AD-A251 642

ATION PAGE

Form Approved OMB No 0704-0188

(2)

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED FINAL 1 Dec 88 - 30 Nov 91
4. TITLE AND SUBTITLE "THE ACQUISITION & UTILIZATION OF SPATIAL & FUNCTIONAL KNOWLEDGE FOR IMAGERY ANALYSIS" (U)		5. FUNDING NUMBERS 62301E 2304/A7
6. AUTHOR(S) Professor David McKeown		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Carnegie-Mellon University Computer Science Division 5000 Forbes Ave. Pittsburgh, PA 15213-3890		8. PERFORMING ORGANIZATION REPORT NUMBER AFOSR-TR- 92 [illegible] 1
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NM Bldg 410 Bolling AFB DC 20332-6448		10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFOSR-89-0199
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; Distribution unlimited		12b. DISTRIBUTION CODE UL

13. ABSTRACT (Maximum 200 words)

In December 1988, researchers at the Digital Mapping Laboratory, School of Computer Science at Carnegie-Mellon University began work on a 30 month contract to explore the acquisition and utilization of spatial and functional knowledge for imagery analysis. Over the course of this grant, they have built on previous research in large-scale knowledge-based systems for the interpretation of aerial imagery. This previous work has focused on the automated analysis of airports and surburban house scenes. Under this grant they have also addressed issues in knowledge acquisition, analysis and evaluation of system performance, and task-level parallelism for large-scale production systems.

14. SUBJECT TERMS			15. NUMBER OF PAGES 40
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR

Computer Science

The Acquisition and Utilization of Spatial and Functional Knowledge for Imagery Analysis

Carnegie Mellon

The Acquisition and Utilization of Spatial and Functional Knowledge for Imagery Analysis

Final Project Report
AFOSR-89-0199
December 1988 through November 1991

Air Force Office of Scientific Research
Building 410
Bolling AFB, D.C. 20332

Dr. Abraham Waksman, COTR
Mathematical and Information Sciences

David M. McKeown, Jr.
Principal Investigator

Digital Mapping Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA. 15213

March 1, 1991

Table of Contents

1. Executive Summary

In December 1988, researchers at the Digital Mapping Laboratory, School of Computer Science at Carnegie Mellon University began work on a 30 month contract to explore the acquisition and utilization of spatial and functional knowledge for imagery analysis as supported under AFOSR Contract AFOSR-89-0199[1] . Over the course of this contract, we have built on previous research in large-scale knowledge-based systems for the interpretation of aerial imagery. This previous work has focused on the automated analysis of airports and suburban house scenes. Under this contract, we have addressed issues in knowledge acquisition, analysis and evaluation of system performance, and task-level parallelism for large-scale production systems.

1.1. Background

The initial research that resulted in the SPAM scene interpretation system was performed between 1984-1986 under funding from the Defense Advanced Research Projects Agency (DARPA) as a part of their program in Image Understanding. It focused on the labeling and recognition of various component structures in aerial imagery containing large-scale commercial and military airports. The goal was to produce a high-level description of scene content using spatial and structural constraints that could be expressed in a rule-based system. At the start of this AFOSR research contract, we had just completed a major reimplementation of the SPAM system, resulting in our ability to use SPAM to interpret suburban housing scenes in addition to airports. This restructuring also enabled us to perform modifications of the SPAM knowledge base using high level descriptions based upon a schema representation language. At this time, the SPAM system included:

- The SPAM rule-based interpretation architecture;
- The RULEGEN compiler and the attribute/value knowledge representation;
- Methods for producing ground truth data, and ground truth databases for each of the 6 airports and 6 suburban housing scenes;
- The SPATS performance analysis tool;
- Several basic knowledge-acquisition tools, including a rule editor and two constraint checking programs (RTFCHK and LCCCHK), which provide the user with graphical feedback about how well a particular first (RTF) or second (LCC) phase constraint performed on ground truth data.

The combination of the rule editor and constraint checking programs proved useful in refining our existing knowledge base, though it made sense to improve the user's ability to simultaneously work with all three tools. This, coupled with the need to easily get at all levels of SPAM's results, inspired the interactive browsing and performance analysis tool SPAMEVALUATE. This tool allows the user to graphically display ground truth data, SPAM interpretations and consistency information, as well as permitting the user to invoke constraints on selected features. Shortly thereafter, a tool to aid in the analysis and improvement of knowledge in the functional-area phase was also added to our suite of knowledge acquisition tools.

Under this AFOSR contract we turned our research focus to the investigation of automatic methods for knowledge acquisition. In our proposal we observed that automated techniques were required and possible for (at least) two reasons:

[1]This research was sponsored by the Air Force Office of Scientific Research under Contract AFOSR-89-0199. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Air Force Office of Scientific Research, or of the United States Government.

- The user must spend a large amount of time examining and testing examples before they can be confident about the addition/modification to the knowledge base. In some cases, this process seemed easy to automate.
- We have available to us our ground truth database, which consists of complete segmentations and annotations for 6 different airports. These are, essentially, "ideal" examples for a learning or knowledge acquisition program.

During the first year of this research contract three preliminary knowledge acquisition tools, RTFANALYZE, LCCANALYZE, and FAANALYZE, were developed to examine the ground truth data and construct rules based on simple statistical measures. The results were mixed, and several problems surfaced, though many of these problems are promising areas for future work. However preliminary, these efforts improved the scene interpretation results generated by SPAM. They also identified the need for the addition of domain independent knowledge to the functional-area phase so that ambiguous interpretations could be resolved.

As we added more knowledge to the system, the end-to-end running time increased. It became apparent that a moderate increase in the size of SPAM's knowledge base would result in a significant increase in the running time; so much so that it would impair our ability to do research. Typical SPAM runs took between 10 to 100 hours on a VAX 11/780. It also seemed clear, from the results of performance analysis, that more knowledge would be necessary to improve SPAM's interpretation results. In order to address the need for more processing efficiency, during the second year we began joint research with members of the Production System Machine group at CMU to explore the utility of task-level parallelism in the context of SPAM. This work produced to several interesting results, including a new SPAM system built on CParaOPS5, a C-based implementation of OPS5, and extensive experience using shared memory multi-processors to achieve near linear speedups.

During the final year we were able to capitalize on our ability to perform larger scale experiments using the CParaOPS5 environment and extend our research on task-level parallelism to including multiple multi-processors communicating using a network shared-memory mechanism. This work also included basic research in model generation and the refinement of our object shape constraints to include relative measurements. In the remainder of this section we describe these accomplishments in more detail.

1.2. Accomplishments

A summary of the research results presented in this report include:

- SPAM was converted from a Lisp-based OPS5 system to CParaOPS5, a new, C-based version of OPS5 created by the Production System Machine group at CMU. In addition to simplifying the overall system design, this change resulted in a 10-fold improvement in execution times (see Section 2.3).

- Our experiences using the new, C-based OPS5 with a large production system such as SPAM resulted in several improvements to the CParaOPS5 system. For SPAM, overall memory usage was reduced to 1/3 of the original CParaOPS5 system (with more significant improvements to pieces of the system). An additional direct benefit to our research was that several large datasets that previously caused SPAM to grow too large are now able to be run (see Section 2.3).

- Several new tools were created for automatic and semi-automatic knowledge acquisition for specific phases of SPAM's processing. These tools were used to augment the first and second phases of SPAM's airport knowledge base (see Sections 3.1, 3.2, 3.3, and 3.4).

- We created a new tool for interactive performance evaluation, called SPAMEVALUATE, which allows the user to interactively browse through the results of a SPAM run (see Section 4.2). Other diagnostic tools were created and/or improved (see Sections 4.1 and 4.3).
- We completed further development of the SPAM image interpretation architecture. Additional domain independent knowledge was added to the third phase of SPAM (functional-area) so that it could more effectively resolve ambiguous (multiple) hypotheses (see Section 5.2.2).
- We learned that limiting the focus of attention within a collection of related features (in our case, within a functional-area) can serve to improve the effectiveness of constraints between those features (see Section 5.2).
- We explored the use of *relative shape constraints* as a new type of knowledge for improving the quality of SPAM's interpretation (see Sections 5.2.3 and 3.4).
- We experimented with three new algorithms for model generation within SPAM (see Section 5.3). The final method, a heuristic search using a domain independent objective function, is currently generating the most correct final airport models (see Section 5.3.3).
- We studied the effectiveness of task-level parallelism on a large production system program (SPAM), achieving near linear speed-ups on a shared-memory multiprocessor (see *Section 6*). We also studied the use of network-shared memory to enable the utilization of two or more networked multiprocessors by parallel applications.

1.3. Publications

The following conference and journal articles, describe research results that were produced under the support of this research contract. In addition a video tape was produced and shown at the DARPA Image Understanding Workshop, held in Pittsburgh, PA. in September, 1990.

1. Harvey, W., Kalp, D., Tambe, M., McKeown, D. and Newell, A. (1991). "The Effectiveness of Task-Level Parallelism for Production Systems" in Journal of Parallel and Distributed Computing, Volume 13, Number 4, December 1991. pp 395-411.

2. Harvey, W., Diamond, M., and McKeown, D. (1990). "The Acquisition and Utilization of Spatial and Functional Knowledge for Image Interpretation", video tape (VHS), 16 minutes and 30 seconds.

3. Harvey, W., Kalp, D., Tambe, M., McKeown, D., and Newell, A. (1990). "The Effectiveness of Task-Level Parallelism for High-Level Vision", *Second ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP),* Seattle, WA. March 14-16, 1990. pp 156-167.

4. Harvey, W., Kalp, D., Tambe, M., McKeown, D. and Newell, A. (1989).

"Measuring the Effectiveness of Task-Level Parallelism for High-Level Vision" in Proceedings of the DARPA Image Understanding Workshop, May 1989. pp 916-933. Also available as CMU Computer Science Technical Report CMU-CS-89-125.

5. D. M. McKeown, Jr., Harvey, W.A., and Wixson, L. (1989). "Automating Knowledge Acquisition For Aerial Image Interpretation" in Computer Vision, Graphics and Image Processing, Number 46, pp 37-81 (1989). Reprinted in Selected Papers on Automatic Object Recognition, Hatem Nasr, Editor. SPIE Milestone Series MS 41, pp. 578-614. (1991)

1.4. Researchers Supported

The following faculty, staff, and graduate students were fully or partially supported under research contract AFOSR-89-0199:

David McKeown is a Senior Research Computer Scientist in the School of Computer Science at Carnegie Mellon University and has been a member of the research faculty since 1986. He received a B.S. degree in Physics and an M.S. degree in Computer Science from Union College, Schenectady, N.Y. Prior to joining the faculty he was a researcher at Carnegie Mellon from 1975, a Research Associate at George Washington University and a member of the Technical Staff at NASA Goddard Space Flight Center, Greenbelt Maryland (1974-1975), and an Instructor in Computer Science and Electrical Engineering at Union College (1972-1974). His research interests in computer science are in the areas of image understanding for remote sensing and cartography, digital mapping and image/map database systems, computer graphics, and artificial intelligence. He is the author of over 35 papers and technical reports and is an active consultant for government and industry in these areas. He is a member of ACM, IEEE, AAAI, American Society for Photogrammetry and Remote Sensing, and Sigma Xi.

Wilson Harvey is a Senior Research Programmer in the School of Computer Science at Carnegie Mellon University. He received a B.S. in Physics and Mathematics from Carnegie Mellon University in 1986. Mr. Harvey has been with the School of Computer Science for more than six years, working primarily on the development of knowledge-based computer vision systems and the representation of spatial knowledge for scene analysis. His research interests include knowledge representation and image understanding.

Milind Tambe is a Research Associate in the School of Computer Science at Carnegie Mellon University. He graduated from Birla Institute of Technology and Science, Pilani, India in 1986 with an M.Sc.(Tech.) in computer science. He completed his PhD in May, 1991 from the School of Computer Science at Carnegie Mellon University. His interests are in the areas of integrated AI architectures and efficiency of AI programs. He is a member of AAAI.

Dirk Kalp is a Senior Research Programmer in the School of Computer Science at Carnegie Mellon University. He graduated from Carnegie Mellon University in 1973 with a B.S. degree in Mathematics and from the University of Pittsburgh in 1981 with an M.S. degree in Computer Science. On the research staff at Carnegie Mellon since 1985, he has worked primarily in the area of parallel production systems. In addition to parallel architectures his research interests include multiprocessor operating systems, virtual memory systems, and the development of tools for fitting symbolic parameter cognitive models.

Matthew T. Diamond is currently a Programmer/Analyst with Shared Medical Systems in Malvern, PA. Formerly, Mr. Diamond was a Research Programmer for the School of Computer Science at Carnegie Mellon University. He received a B.S. degree in Mathematics (Computer Science track) from Carnegie Mellon University in May 1989. His research interests in computer science are in the areas of image understanding for aerial photo-interpretation, knowledge acquisition, and computer graphics. Mr. Diamond is a member of ACM.

Anurag Acharya obtained his B.Tech (Computer Sc and Engg.) in 1987 from IIT Kharagpur, India. He is currently a doctoral candidate at the School of Computer Science, Carnegie Mellon University. His research interests are production systems, programming languages and parallel processing.

In the body of this final report we provide a detailed description our work with the SPAM image interpretation system from December 1988 through November 1991 as supported under contract AFOSR-89-0199. Section 2 provides a background discussion of the overall rule-based image interpretation system. In Section 2.3 we discuss changes to the underlying SPAM architecture necessary to support knowledge acquisition and performance evaluation. We relate these revisions and additions to our knowledge acquisition and analysis tools in Sections 3 and 4, respectively. Experimental results from each phase of processing in SPAM are presented and discussed in Section 5, while Section 6 contains a discussion of our work with task-level parallelism for high-level computer vision systems. Finally, we describe our current work and present our ideas for future research (Section 7). We believe that progress has been steady and that the work in knowledge acquisition and system analysis has greatly improved our understanding of fundamental research issues in knowledge-based computer vision.

2. The SPAM Architecture

SPAM is a production system architecture for the interpretation of aerial imagery with applications to automated cartography and digital mapping [1, 2]. It tests the hypothesis that the interpretation of aerial imagery requires substantial knowledge about the scene under consideration. Knowledge about the type of scene, whether airport, suburban housing development, or urban city, aids in low-level and intermediate level image analysis, and will drive high-level interpretation by constraining search for plausible consistent scene models. SPAM has been applied in two task areas: airport and suburban house scene analysis. In this section we describe the SPAM architecture, then briefly discuss related work in knowledge acquisition and knowledge-based vision.

2.1. Background: The SPAM Architecture

As with many computer vision systems, SPAM attempts to interpret the 2-dimensional image of a 3-dimensional scene. A typical input image is shown in Figure 1. The particular goal of the SPAM system is to interpret an image segmentation, composed of image regions, as a collection of real-world objects. For example, the output for the image in Figure 1 would be a model of the airport scene, describing where the runway, taxiways, terminal-building(s), etc., are individually located. SPAM uses four basic types of scene interpretation primitives: *regions*, *fragments*, *functional-areas*, and *models*. SPAM performs scene interpretation by transforming image *regions* into scene *fragment* interpretations. It then aggregates these fragments into consistent and compatible collections called *functional-areas*. Finally, it selects sets of functional-areas to form *models* of the scene.

As shown in Figure 2, each interpretation phase is executed in the order given. SPAM drives from a local, low-level set of interpretations to a more global, high-level, scene interpretation. There is a set of hard-wired productions for each phase that control the order of rule executions, the forking of processes, and other domain-independent tasks. However, this "bottom-up" organization does not preclude interactions between phases. For example, prediction of a fragment interpretation in *functional-area* (FA) phase will automatically cause SPAM to reenter *local-consistency check* (LCC) phase for that fragment. Other forms of top-down activity include stereo verification to disambiguate conflicting hypotheses in *model-generation* (MODEL) phase and to perform linear alignment in *region-to-fragment* (RTF) phase.

The first phase of SPAM is called region-to-fragment (RTF). This is a traditional heuristic classification process, using knowledge about the classes of features that occur in the scene to map a segmentation to a set of interpretations. Local properties of the segmentation, such as shape, texture, or height, are used to decide on which interpretations to generate. Examples of the type of knowledge used in region-to-fragment would be *runways are typically 50 to 80 meters wide*, or *houses are 8 to 10 meters high*.

The second phase of SPAM is called local-consistency check (LCC). This phase performs a modified constraint satisfaction between the interpretations generated in region-to-fragment. The knowledge in this phase consists of the constraints between the different classes of objects. An example constraint would be *runways have perpendicular taxiways*. There are many such constraints in the system, with each constraint providing weak support to the participating hypotheses. Thus, it is not the action of a single constraint, but the collective action of several constraints which causes two interpretations to be found consistent with one another.

The functional-area (FA) phase groups together those interpretations that support one another, where support is computed from the results of the previous phase. A functional-area is defined

Figure 1: Aerial image of San Francisco Airport.

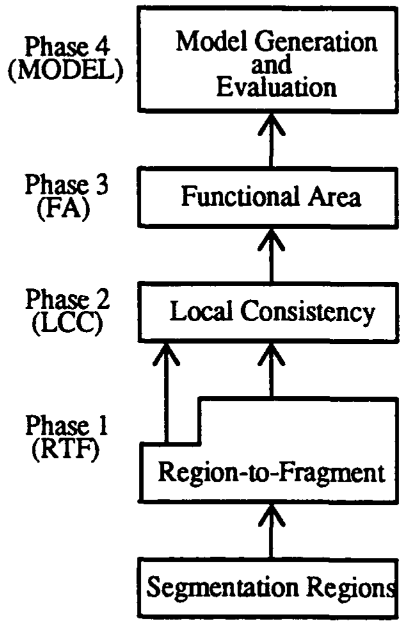


Figure 2: Interpretation phases in SPAM.

as a group of interpretations that are similar in function and are in close physical proximity to one another. For instance, a terminal functional-area is defined to contain only terminal-building, parking-lot, road, and parking-apron interpretations. It is physically represented by the convex hull of the features in the functional-area.

Finally, the model-generation (MODEL) phase uses the functional-areas and combines them based on a number of heuristics, including number of conflicts, number of supported interpretations, and area of coverage. Conflicts are identified and resolved, either using support within the context of the model, or by invoking some process which would provide additional knowledge. For instance, if in the context of a model a region of the image was interpreted as both a taxiway and a hangar-building, a stereo process could be invoked to help resolve the conflict based on the region's height estimate. Multiple models are commonly generated and these models can be used as contexts for further processing.

2.2. Related Work

There has been a large body of work in traditional knowledge-based systems to support knowledge acquisition and validation [3]. Typical issues include the validation of knowledge-based systems, checking for completeness and consistency in knowledge bases, and system performance evaluation. The application areas are broad, spanning medical diagnosis, design, process control, and configuration analysis. However, little work in this area has been focused within the context of computer vision tasks. One explanation is that there are few end-to-end knowledge-based systems for computer vision in the literature. A corollary is that those knowledge-based systems that have been built are "one-person" systems that do not survive much past the thesis defense. Some of the earliest "high-level" vision systems applied to aerial image analysis include Matsuyama's system [4], the Sigma system [5], and SPAM [1]. More recently, researchers have explored the use of generic knowledge [6] and investigated the formalization of knowledge to support high-level vision [7].

Our previous work in this area [8] focused on the use of compilation tools to translate a high-level schema based constraint representation into OPS5 productions. We also developed static analysis tools to aid in the display and debugging of the model descriptions generated by SPAM. As a result of the long term use of this system, our research focus has shifted toward issues in automating portions of the knowledge acquisition process and in the development of tools to aid in the diagnostic analysis of finer levels of system behavior. After a description of some of our previous work, we will discuss our recent work in these areas.

2.2.1. RULEGEN

The SPAM architecture was originally hand-coded in OPS5 for airport scene interpretation. However, hand-coding such a system for several different scene domains, not to mention modifying and maintaining existing systems, is a formidable task. Breaking the system into domain-independent and domain-dependent parts was an important research goal. RULEGEN is a knowledge compiler that takes as input an intermediate knowledge representation which is schema based and automatically generates OPS5 productions. Domain independent OPS5 productions control the execution of the generated rules.

```
@BEGIN taxiway
'CLASS' = 'taxiway'
'REGION-DEPENDENCES' = ''
'FRAG-DEPENDENCES'  = ''
'SHAPE-CONSTRAINT'  = 'curvature && 0.000000 <= value <= 0.250628'
'SHAPE-CONSTRAINT'  = 'area && 0.000000 <= value <= 62394.166614'
'SHAPE-CONSTRAINT'  = 'perimeter && 0.000000 <= value <= 3310.868967'
'SHAPE-CONSTRAINT'  = 'fractional-fill && 0.033127 <= value <= 0.686500'
'SHAPE-CONSTRAINT'  = 'ellipse-length && 50.692725 <= value <= 1253.353940'
'SHAPE-CONSTRAINT'  = 'ellipse-width && 13.489274 <= value <= 117.718124'
'SHAPE-CONSTRAINT'  = 'ellipse-linearity && 0.000000 <= value <= 34.161718'
'SHAPE-CONSTRAINT'  = 'compactness && 0.000671 <= value <= 0.046449'
'SHAPE-CONSTRAINT'  = 'orientation && 0.000000 <= value <= 3.671971'
'SHAPE-CONSTRAINT'  = 'mbr-length && 0.000000 <= value <= 1458.611861'
'SHAPE-CONSTRAINT'  = 'mbr-width && 0.000000 <= value <= 533.859851'
'SHAPE-CONSTRAINT'  = 'mbr-linearity && 0.000000 <= value <= 52.576719'
@END taxiway
```

Figure 3: An example schema from the first phase of SPAM.

An example of our knowledge representation format for the first phase of SPAM is shown in Figure 3. Knowledge about the local properties of the segmentation are represented in these schema. This schema defines taxiway objects to be 50 to 1200 meters in length, 13 to 117 meters wide, et cetera. The schema information is automatically compiled into the set of OPS5 productions shown in Figure 4. RULEGEN produces SPAM control productions to initialize the ruleset, and one production for each of the attributes to be tested. Domain independent rules take the results from these production firings and evaluate whether or not a runway interpretation should be generated for this segmentation.

For the other phases of SPAM, a similar compilation process occurs, though the schema attributes must change. In the second phase, for example, constraints between interpretation classes must be represented. Schema files for the second phase of SPAM encode the participating hypotheses, bounds for the geometric tests, and a confidence value. Each set of attribute/value pairs generates eight or more OPS5 productions to do initialization, clean-up, iteration, low-level computations, and result evaluation.

This separation of the domain-dependent portion of the system from the domain-independent

```
;***************************************************************************
; **** Beginning of taxiway attribute-matches ****
;***************************************************************************

(p RTF--TW--initialize-TW-attributes
  (rtf-task ^region <name> ^data <token>)
  (region ^symbolic-name <name> ^taxiway nil)
 -->
  (make rtf-subtask ^ruleset TW--match-TW-attributes
   ^region <name> ^data <token> taxiway)
)

(p RTF--TW--match-TW-curvature
  (rtf-subtask ^ruleset { <ruleset> = TW--match-TW-attributes }
   ^region <name> ^data { } <hyp>)
 { (rtf-rule-constants ^ruleset <ruleset>
   ^attribute curvature) <constants> }
  (region ^symbolic-name <name> ^curvature <value>)
 -->
  (bind <index> (litval constants))
  (call OPS_match_score <name> <hyp> <value>
   (substr <constants> <index> inf))
)

(p RTF--TW--match-TW-area
  (rtf-subtask ^ruleset { <ruleset> = TW--match-TW-attributes }
   ^region <name> ^data { } <hyp>)
 { (rtf-rule-constants ^ruleset <ruleset>
   ^attribute area) <constants> }
  (region ^symbolic-name <name> ^area <value>)
 -->
  (bind <index> (litval constants))
  (call OPS_match_score <name> <hyp> <value>
   (substr <constants> <index> inf))
)

...
```

Figure 4: OPS5 productions generated from the schema in Figure 3.

portion impacts SPAM greatly. With over 600 productions in the current system, automatic compilation not only makes engineering simpler, it also allows greater freedom for knowledge acquisition. The knowledge acquisition process is not tied to OPS5, nor does it have to understand the inner workings of the SPAM architecture.

2.3. Architectural and Engineering Changes

Several modifications have been made to the domain-independent productions that comprise the SPAM architecture. These have been made solely in the last two phases of the system, the functional-area and model-generation phases. In the functional-area phase, more domain-independent productions were added to implement resolution of ambiguous interpretations within the context of single functional-areas. The model-generation phase is currently being modified to allow more knowledge to be used in the search for consistent functional-areas. This work is discussed in detail in Sections 5.2 and 5.3, respectively.

In addition to the architectural changes, we have, as a result of our research into task-level parallelism[2], moved SPAM from a Lisp-based OPS5 to a new system called CParaOPS5. This system, produced by the Production System Machine group here at CMU, is a full implementation of OPS5, with the additional capability of performing the match in parallel. CParaOPS5 is written entirely in C, and its rule compiler generates C-code. Even without using the parallel matching capabilities, the conversion has yielded an approximate 10-fold increase in performance, and has allowed us to port SPAM to several different hardware architectures, including the Encore Multimax, the DEC 3100, and the Sun SPARCstation.

Our use of CParaOPS5 has not only impacted our research, but it has resulted in several improvements in the CParaOPS5 system, as well. Some of these improvements are:

Working-memory element size	The original implementation used a static working-memory element (wme) size of 128 attributes. The new implementation dynamically allocates wmes whose sizes are computed from the number of declared attributes. The relative amount of memory used for wmes in the new system is 15-25% of what the original system used.
Conflict set	A new representation allowed the size of the conflict set to shrink to 10% of it's original size. The algorithm was also improved, resulting in a 10% overall speedup.
Code optimizations	There were several places in the original CParaOPS5 system where small (but significant) optimizations could easily be implemented. These included the pre-computing of common subexpressions, register allocation, and moving values from global to local variables. The net performance gain was another 10%.
Statistics on resource usage	The CParaOPS5 system is now instrumented so that a user can get end-run statistics on every aspect of the system. Some examples are wme-memory usage, alpha and beta memory usage, symbol-table size, et cetera.

Minor changes to the OPS5 language semantics were required for these changes to be possible. However, the entire CParaOPS5 system has been engineered so that the user can compile it for flexibility (using the original OPS5 semantics) or efficiency (using the new semantics). For SPAM, these changes resulted in an overall 5-10% improvement in running times, and a 66% improvement in memory usage. More importantly, however, the changes allowed us to run SPAM on much larger datasets which we were previously unable to run.

[2]See Section 6 for details.

3. Issues in Knowledge Acquisition for SPAM

In our previous work in knowledge compilation SPAM was separated into domain-dependent and independent parts to facilitate generalization, as well as to aid in our ability to maintain the system. The system is generated from a knowledge representation that is not dependent on the implementation language (which, in the current SPAM system, is OPS5). With the knowledge-base decoupled from OPS5, it is easier to create and maintain the large knowledge-bases needed to analyze complex scenes.

Phase	Tool(s)	Kind(s) of Knowledge	Difficulties Encountered Using this Knowledge
RTF	RTFCHK, RTFANALYZE, FAANALYZE	Shape	Difficult to visualize using attribute/value representation; Large variations across scenes; Can require many examples; Shapes of object classes aren't unique.
LCC	LCCCHK, FAANALYZE	Local geometric	Context dependent; Many weak constraints required.
FA	FAANALYZE, SPAMEVALUATE	Grouping	Subjective functional definitions somewhat ambiguous; Very dependent on knowledge in LCC.
MODEL	SPAMEVALUATE	Global geometric, Conflict	Many knowledge sources (from different levels of processing) are required.

Table 1: Types of Knowledge Used in SPAM.

Tool	Phase(s)	Interaction	Problems Addressed
RTFCHK	RTF	Textual, Graphical	Allows graphical evaluation of shape constraints.
RTFANALYZE	RTF	Textual	Compiles shape constraints from examples.
LCCCHK	LCC	Textual, Graphical	Allows graphical evaluation of local geometric constraints.
FAANALYZE	RTF, LCC, FA	Graphical, Textual	Generates shape and local geometric constraints from examples using contexts
SPATS	All	Textual	Provides statistical feedback on SPAM's performance
SPAMEVALUATE	All	Graphical, Textual, Static graphical	Allows interactive graphical and textual access to all phases of SPAM results.

Table 2: Summary of Tools for Knowledge Acquisition in SPAM.

Tables 1 and 2 summarize the types of knowledge used in SPAM, the problems encountered in acquiring and using this knowledge, and the tools developed to address these problems. Several tools appear to be missing (e.g., FACHK). These may be added as necessary as the development of this line of research continues. Each of the currently implemented tools will be discussed in detail in the following sections.

3.1. Knowledge Acquisition of Spatial and Structural Constraints

RTFCHK, a knowledge acquisition tool for the first phase of SPAM, allows the user to interactively study and modify schemas encoding SPAM's shape knowledge. Correctness of individual rules is determined by comparing their results against the ground-truth interpretations of a scene. The results are sorted by correctness and by pass/fail into four displays. Figure 5 shows a confusion matrix generated from applying a runway rule to data from Washington National Airport. The matrix display shows true positives in the upper left, false negatives in the upper right, false positives in the lower left, and true negatives in the lower right. A rule containing constraints that are too restrictive will result in many false negatives, while loose constraints will cause many false positives. A "perfect rule" will have no false positives or negatives. Upon viewing these results the user may edit the schema and test it again.

RTFCHK was employed to improve a set of RTF schema that had been generated automatically from ground-truth files. The result was a set of RTF schema that exhibited improved performance over all of our airport test scenes (see discussion in Section 5.1). It is accepted that shape-classification alone cannot accurately classify complex scenes, but the RTF schemas produced using this tool provide better sets of interpretations for the later phases of SPAM.

A tool very similar to RTFCHK, called LCCCHK, aids the user in the knowledge acquisition process for the second phase of SPAM. For this phase, pairs of regions are tested against one another for attributes like closeness and perpendicularity. LCCCHK allows single constraints to be applied to classes of objects, and the results are compared against the ground-truth. Again, the user is presented with a confusion matrix display from which they may interactively inspect the results and/or make modifications to the knowledge base.

3.2. Automatic Shape Constraint Generation

One way for a user to develop new RTF schema is to allow them to measure many example regions. As more examples are viewed, the user can confidently assign reasonable values for the shape-attribute constraints. Given that we have generated several ground-truth datasets containing many examples of every class, this process can be automated.

RTFANALYZE is a tool that creates new RTF schema when given one or more scenes with ground-truth. It statistically analyzes the associated ground-truth segmentation to produce values, by class, for the various shape attributes. Currently, the process is exhaustive in that every attribute is computed for every class.

Given sufficient examples, RTFANALYZE produces schema that perform well across a range of airports[3]. The automatic schema generator produces effective rules much faster than a user can

[3]Because the process is statistics based, this qualifier depends on how representative the examples are of the target class. Of course, other machine learning techniques, such as decision trees or clustering, can be used to help solve this problem.

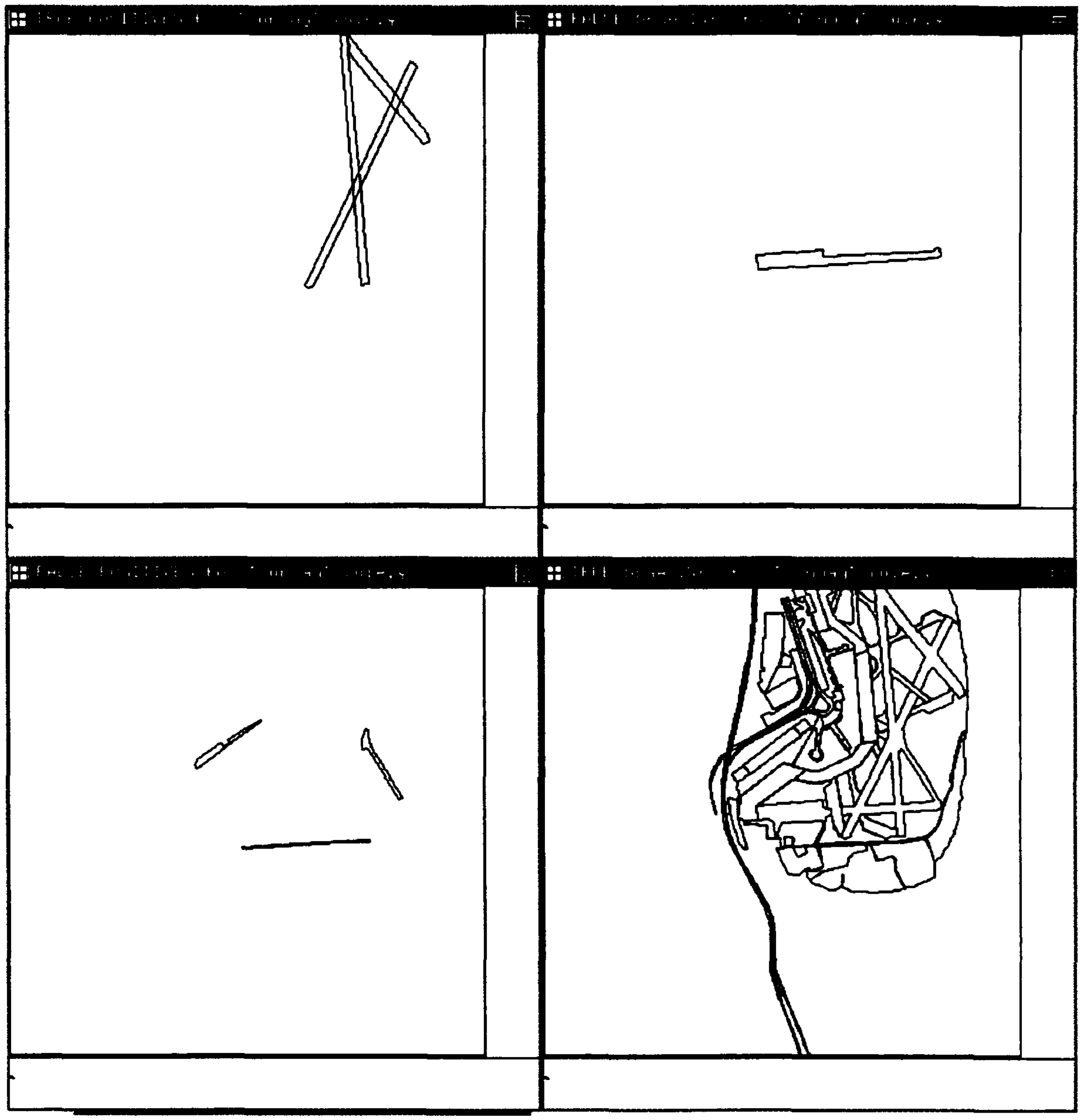


Figure 5: RTFCHK display after single schema has fired on ground-truth segmentation from Washington National Airport.

interactively, but often manual adjustments will improve a rule's performance. The combination of automatic generation followed by manual adjustment is currently our normal mode of operation for producing RTF schema.

3.3. Semi-Automatic Structural Constraint Generation via Functional Ground Truth

Much of SPAM's effort in hypothesis refinement is performed in the LCC phase. Because the space of possible constraints is large, manual methods of knowledge acquisition are inadequate.

Automatic or semi-automatic methods for augmenting LCC knowledge are not only useful, but they become necessary to ensure that the complexity of the scene to be interpreted is captured.

The combinatorics of the second phase make exhaustive observation of example scenes a very lengthy process, as every pair of regions must be tested using every constraint across several data sets. Furthermore, the LCC constraints would be extrapolated based on the observed relationships between arbitrary pairs of regions. This makes little sense when regions are unrelated; for example, a taxiway is usually perpendicular to an adjacent runway, but has no consistent relationship to distant runways.

In order to reduce the combinatorics and to be more precise about what context is used in the analysis, we decided to analyze only the structural relationships between elements of each functional-area type. This assumes that interactions between objects of different functional-areas is weak. The schema produced in this manner make more sense because they are derived from local relationships. Generating LCC schema automatically requires example (ground-truth) functional-areas.

FAANALYZE is a tool for semi-automatically acquiring and analyzing ground-truth functional-areas. It has an interactive mode that allows the user to specify a functional-area by clicking on its constituent regions. The system only allows the user to add regions which belong to the correct functional-area type. For example, when entering a hangar functional-area, the user may choose only hangar buildings, roads, grassy-areas, and parking-aprons. Figure 6 is a snapshot of the display as a user adds regions to an example hangar functional-area.

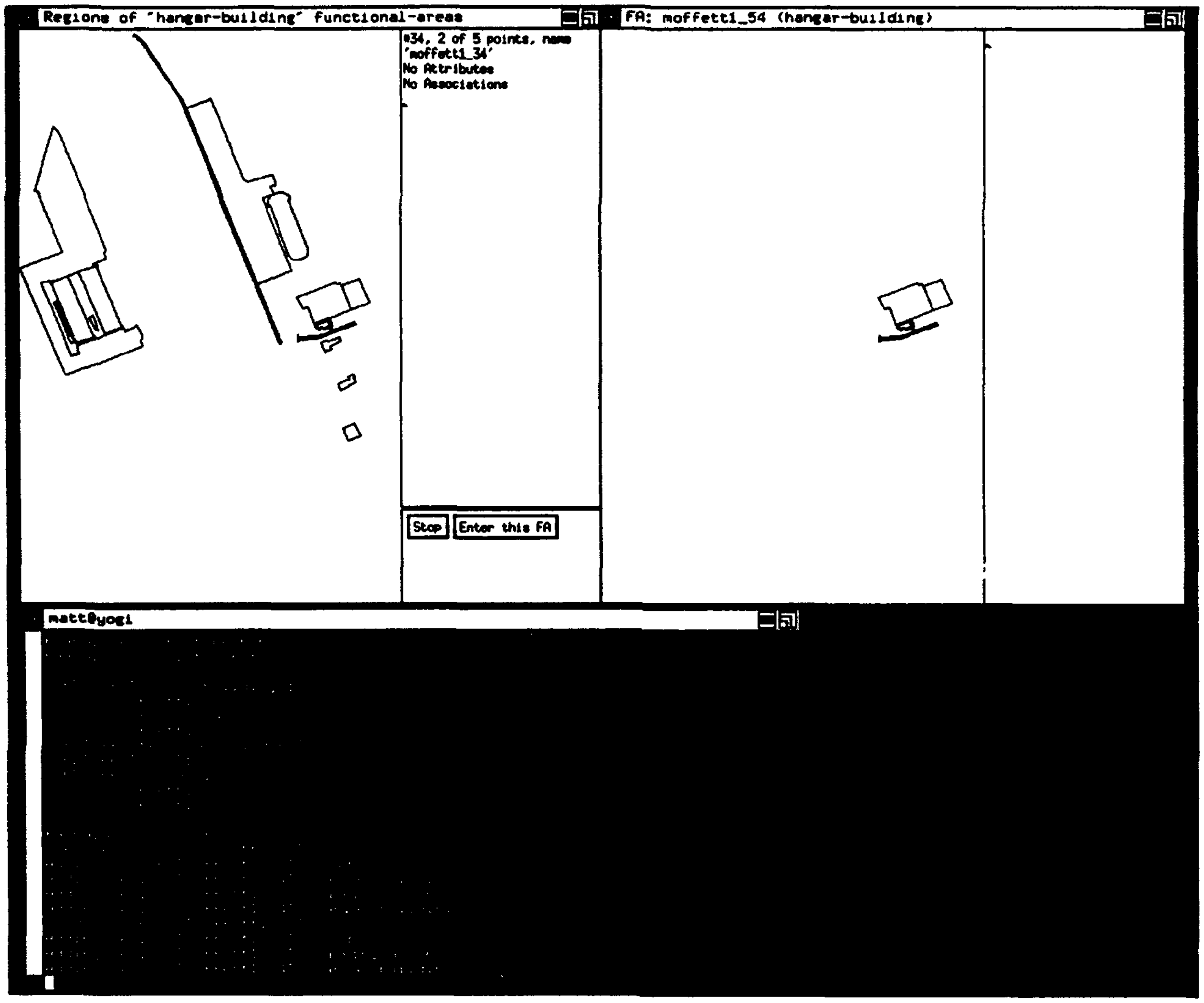
Once ground-truth functional-areas have been created, FAANALYZE performs a statistical analysis, recording the observed values of the LCC constraints. Choosing which constraint types (distance, orientation) to use is a difficult problem, and is currently performed by the user. We are investigating the use of observed geometric distributions to decide which constraints perform best. As discovered with the RTF phase, a manual post-processing phase helps to improve the quality of the constraints generated automatically.

3.4. Acquiring Relative Shape Constraints

The program FAANALYZE addresses two issues. First, we wanted to generate LCC schema automatically by observing the relationships between region-types within functional-areas, as discussed previously. Second, we wanted to explore *relative shape constraints*.

We define relative shape attributes as the ratio between an attribute of functional-area seed-region and the same attribute for an element of the functional-area. For instance, we hypothesized that if the seed region was larger than average, then one might expect the elements around it to be larger, as well. Several working-memory elements representing relative shape constraints for hangar-buildings and runways are shown in Figure 7. Unlike the first phase of SPAM, which tests the absolute shape-attributes of regions, the relative constraints take into account the relative size of the airport or subsets of it. The numbers, therefore, are unitless.

Generating relative shape constraints and LCC schema automatically requires example functional-areas. Once sample functional-areas have been created, FAANALYZE analyzes them, recording the observed values of both the relative shape constraints and the LCC geometrics. The output is in the form of working-memory elements and LCC schema. These are used to generate a new SPAM system.


Regions of 'hangar-building' functional-areas
FA: moffett1_54 (hangar-building)
Stop
Enter this FA
matt@yogi

```
#34, 2 of 5 points, name
'moffett1_34'
No Attributes
No Associations
```

Figure 6: FAANALYZE: Selecting regions to build an example functional-area.

```
(make fa-constraint  ^fa-context hangar-building
   ^from-class hangar-building  ^to-class road
   ^attribute  area        ^minimum 0  ^maximum 1818693   ^average 442)
(make fa-constraint  ^fa-context hangar-building
   ^from-class hangar-building  ^to-class tarmac
   ^attribute  perimeter  ^minimum 0  ^maximum 73371    ^average 728)

(make fa-constraint ^fa-context runway
  ^from-class runway           ^to-class taxiway
  ^attribute  area        ^minimum 1  ^maximum 157      ^average 9)
(make fa-constraint  ^fa-context runway
   ^from-class runway           ^to-class taxiway
   ^attribute perimeter   ^minimum 4  ^maximum 170      ^average 16)
```

Figure 7: Example relative shape constraints.

The relative shape constraints can be used in the following three ways:

- To filter incorrect hypotheses from each functional-area;
- As a measure of confidence for entire functional-areas; if the ratios are outside the expected ranges, then either the seed region of the functional-area is incorrect or many constituents of the functional-area are wrong;
- To filter out unlikely hypotheses *before* the LCC phase fires, reducing the number of competing hypotheses in the LCC phase.

The relative shape constraints do, in fact, filter out hypotheses from functional-areas. However, we found that the local-consistency phase also filters many of the same hypotheses. We investigated using the relative shape-constraints to evaluate functional-areas as a whole, but the difference between a correct and incorrect functional-area was found to be difficult to quantify. This leaves only the third use, for filtering out hypotheses before the LCC phase fires. Because these constraints almost duplicate a subset of the LCC phase's results, we could use the relative shape-constraints without changing the final results. Since relative shape-constraints are faster than the LCC geometrics, this should reduce SPAM's total running time.

4. Analysis and Evaluation Tools

Post-run analysis and evaluation provides important feedback about the correctness of the knowledge-base and ways in which it can be improved. Some of our older tools, such as SPATS [8], produce statistical summaries of the performance of the SPAM system. Static run analysis can provide good summaries of system performance, allowing a user to easily gauge the aggregate affects of modifying a system's inputs. However, it can be difficult reasoning from this analysis about the specifics of why a result came to be. For example, knowing that a certain number of taxiways failed to pass a particular rule provides no information as to which taxiways failed, or why. While there is currently no automatic diagnostic tool for a SPAM user to consult, it seemed most natural to provide a graphical tool that would allow the user to interact with the results from a SPAM run. Our experience has found SPAMEVALUATE to be an effective diagnostic tool.

4.1. Statistical Performance Evaluation

SPATS is an older utility that generates performance statistics on SPAM runs. These statistics are used to chart improvements in SPAM's performance and to expose weaknesses. Recently we made some simple (but important) improvements in the output format to make it easier to use and understand. Some of these include condensing the generated tables, improving the statistics by breaking them into categories, and including explanations of output directly with the statistics.

We also rewrote the analysis code to conform with the changes we made in converting SPAM to CParaOPS5. The code that SPATS used in accessing working-memory element dumps was slow and memory inefficient. This seriously hindered (and sometimes prevented) the analysis of large SPAM runs, so we rewrote it to require less storage in memory. It also performs many times faster than before, and with some added functionality. Other programs now use this code as well, including SPAMEVALUATE.

4.2. Interactive Performance Analysis

SPAMEVALUATE provides an interactive medium for browsing SPAM results and evaluating proposed modifications. It generates run statistics that augment those already generated by SPATS. It produces exhaustive and summary output. *Dumps* are a human-readable reproduction of the results of each phase; *summaries* are statistical synopses of the results. For example, the dump of the RTF phase lists each region, along with every RTF schema it passed. The summary, on the other hand, is a chart showing the breakdown of correct interpretations versus incorrect, allowing the user to infer which interpretations tend to be mistaken for one another. Statistics are available for the first 3 phases of SPAM; a sampling is shown in Figure 8.

Using the graphical mode of SPAMEVALUATE, the user can select regions by clicking with a mouse and then display very specific information culled from the SPAM run. The user can jump from phase to phase, observing how the results from each of the phases interact. The Figure 9 shows, in overlapping windows, the various phases of results as the user displays each in turn. Since the results are displayed graphically, the user gets a more intuitive feel for the meaning of the results. For example, a statistical summary might show that many taxiways are failing, but the graphical display would show that the failing taxiways tend to be longer than average, implying that the problem might lie in the RTF phase.

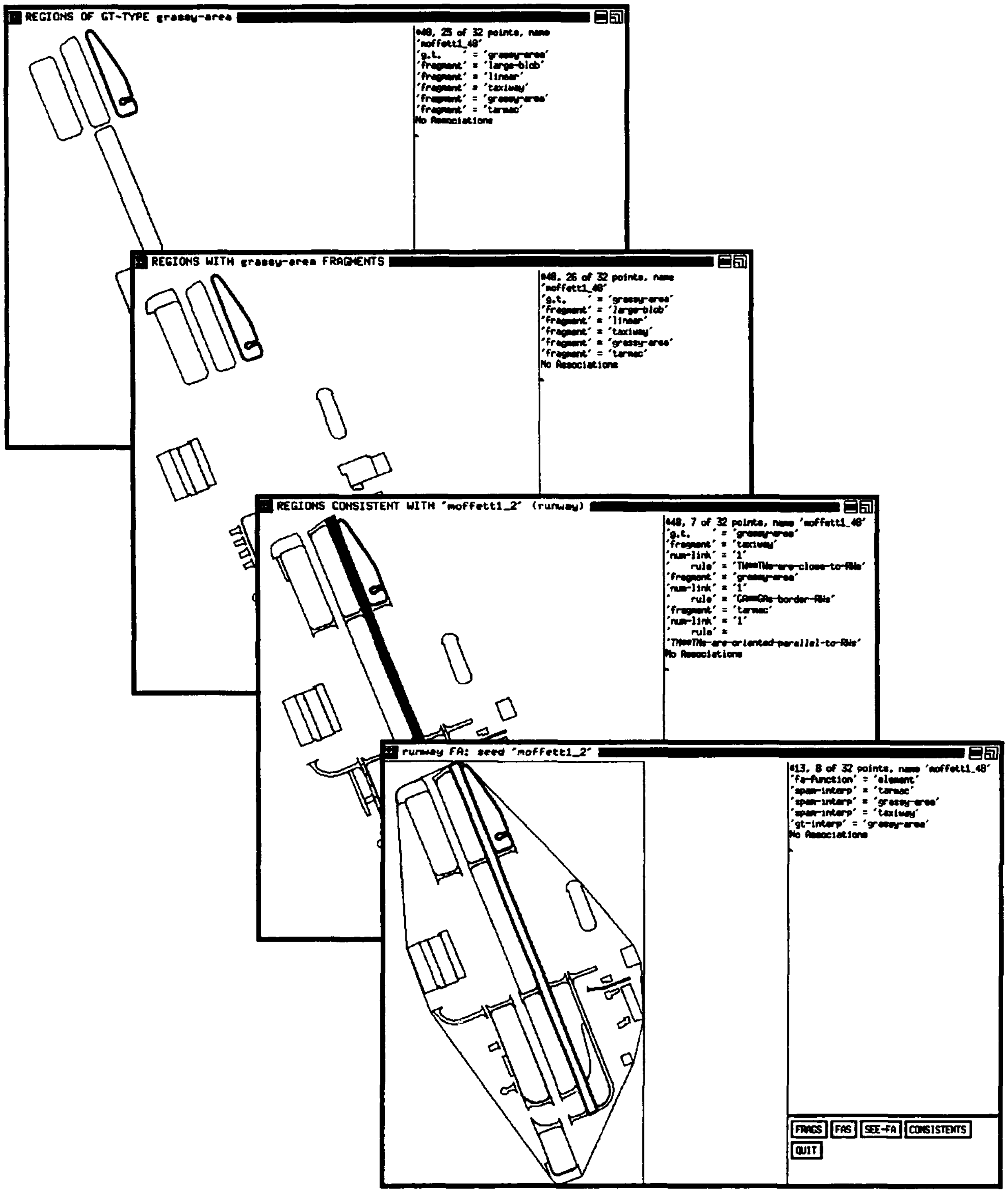
Another feature of SPAMEVALUATE is the ability to invoke schema of the first and second phases on individual regions, similar to the functionality provided by RTFCHK and LCCCHK. Since

```
Generic Image: moffett1
Scenetype: airport
[RTF DUMP: interpretations generated by ground-truth region id]
    moffett1_0: navigational-aid parking-apron parking-lot grassy-area
                tarmac maintenance-building taxiway terminal-building
    moffett1_1: parking-lot taxiway road terminal-building
    moffett1_2: runway
        ...
[RTF STATS: class confusion matrix]
G.T. INTERPRETATION
|  SPAM INTERPRETATION -->
V       RW   TW  MRD  ARD HWRD   CB  CHB   HB   MB   TB   CT   PA   PL   GA   TM  PMR  NVA  APJ
  RW     2    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  TW     0   36    0   15    0    0    0   10   15   27    0   16   28   15   17    0    8    0
 ARD     0    1    0    2    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  HB     0    5    0    1    0    1    0    9    7    7    0    6    7    7    7    0    6    0
  MB     0    1    0    0    0    0    0    0    1    1    0    1    1    1    1    0    1    0
  TB     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  PA     0    0    0    0    0    0    0    0    1    1    0    3    0    1    2    0    1    0
  PL     0    3    0    1    0    0    0    0    2    2    0    2    3    2    2    0    0    0
  GA     0    1    0    0    0    0    0    0    1    1    0    4    1   11    8    0    2    0
      ...
[LCC STATS: pairwise consistency table]
             HYPOTHESIS-PAIR (t-t)c (t-t)i (t-f)c (t-f)i (f-f)c (f-f)i Total#
     taxiway--runway           92%    ---    ---     1%    ---     0%     64
        road--road              2%    ---    ---    12%    ---    72%     50
hangar-buildi--road             4%    ---    ---    10%    ---    63%    108
hangar-buildi--hangar-buildi   21%    ---    ---     2%    ---    69%     92
terminal-buil--road             0%    ---    ---    10%    ---    85%    203
parking-apron--hangar-buildi    7%    ---    ---    13%    ---    67%    188
      ...
[FA DUMP: functional-area constituents by type]
FUNCAREA_77 (terminal)-- moffett1_1[TW]
         road: moffett1_32*
  parking-apr: moffett1_48[GA] moffett1_47[GA]
  parking-lot: moffett1_0[TW]
FUNCAREA_76 (road)-- moffett1_1[TW]
  grassy-area: moffett1_0[TW] moffett1_48* moffett1_47*
      ...
[FA STATS: comparison of generated interpretations to ground-truth by FA type]
runway
        runway-- 2 of 2 correct
        taxiway-- 21 of 21 correct
        grassy-area-- 11 of 11 correct
        tarmac-- 0 of 1 correct (1 GA)
hangar
        road-- 4 of 26 correct (18 TW) (4 PL)
        hangar-building-- 28 of 97 correct (68 TW) (1 TM)
        parking-apron-- 8 of 49 correct (25 TW) (8 HB) (7 PL) (1 GA)
        tarmac-- 0 of 21 correct (3 MB) (18 GA)
      ...
```

Figure 8: Selected statistics generated by SPAMEVALUATE.

tracing through the results often suggests possible deficiencies in the current ruleset, the user can test such conclusions by invoking individual schema on any chosen region. The rule fires in a verbose mode, showing which constraints passed or failed, and by how much, as seen in Figure 10.

In practice, SPAMEVALUATE has been invaluable in our evaluation of the SPAM system. Sometimes a problem is solved by adjusting a constraint, or by adding a new schema to express a new piece of knowledge. SPAMEVALUATE also facilitates access to information necessary to study new methods of processing in the (more complex) functional-area and model phases. It is useful to be able to reason backwards from an interpretation result containing an error to the specific piece of knowledge causing that error. For instance, we recently changed the FA phase to remove competing hypotheses from within a functional-area, resulting in unique labels for the constituent objects (see Section 5.2.2). SPAMEVALUATE aided the algorithm development by permitting us to interactively trace back relationships from sets of functional-areas to LCC constraints between specific pairs of objects.


REGIONS OF GT-TYPE grassy-area
REGIONS WITH grassy-area FRAGMENTS
REGIONS CONSISTENT WITH 'moffett1_2' (runway)
runway FA: seed 'moffett1_2'
FRAGS
FAS
SEE-FA
CONSISTENTS
QUIT

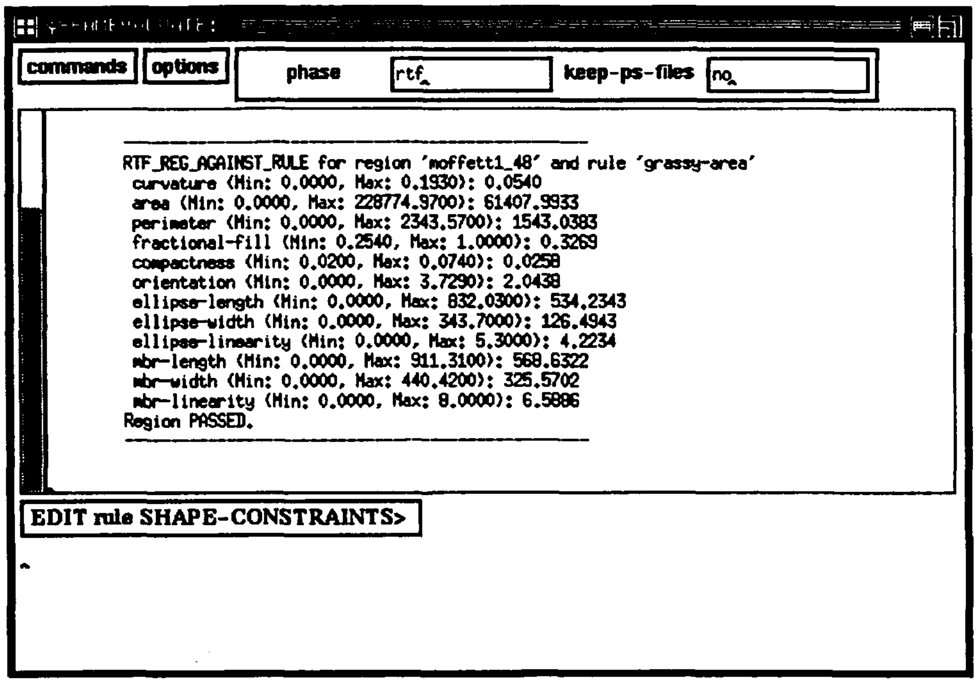
```
#48, 25 of 32 points, name
'moffett1_48'
'g.t.      ' = 'grassy-area'
'fragment' = 'large-blob'
'fragment' = 'linear'
'fragment' = 'taxiway'
'fragment' = 'grassy-area'
'fragment' = 'tarmac'
No Associations
```

```
#48, 26 of 32 points, name
'moffett1_48'
'g.t.      ' = 'grassy-area'
'fragment' = 'large-blob'
'fragment' = 'linear'
'fragment' = 'taxiway'
'fragment' = 'grassy-area'
'fragment' = 'tarmac'
No Associations
```

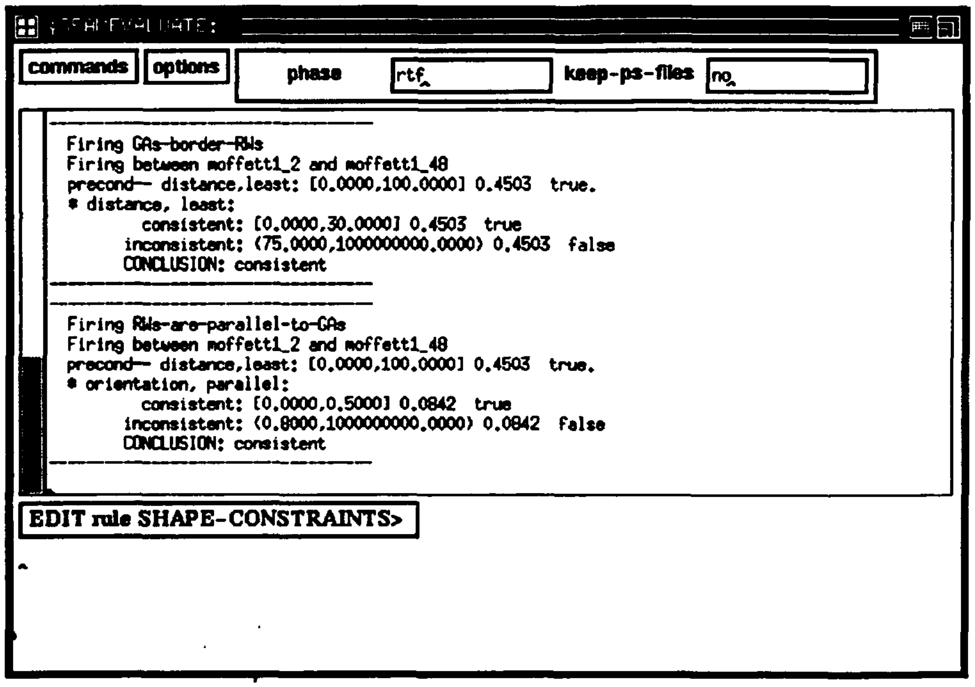
```
#48, 7 of 32 points, name 'moffett1_48'
'g.t.      ' = 'grassy-area'
'fragment' = 'taxiway'
'num-link' = '1'
'     rule' = 'TW**TWs-are-close-to-RWs'
'fragment' = 'grassy-area'
'num-link' = '1'
'     rule' = 'GA**GAs-border-RWs'
'fragment' = 'tarmac'
'num-link' = '1'
'     rule' =
'TW**TWs-are-oriented-parallel-to-RWs'
No Associations
```

```
#13, 8 of 32 points, name 'moffett1_48'
'fa-function' = 'element'
'spam-interp' = 'tarmac'
'spam-interp' = 'grassy-area'
'spam-interp' = 'taxiway'
'gt-interp' = 'grassy-area'
No Associations
```

Figure 9: Various evaluation methods in SPAMEVALUATE.



```
------------------------------------------------
RTF_REG_AGAINST_RULE for region 'moffett1_48' and rule 'grassy-area'
 curvature (Min: 0.0000, Max: 0.1930): 0.0540
 area (Min: 0.0000, Max: 228774.9700): 61407.9933
 perimeter (Min: 0.0000, Max: 2343.5700): 1543.0383
 fractional-fill (Min: 0.2540, Max: 1.0000): 0.3269
 compactness (Min: 0.0200, Max: 0.0740): 0.0258
 orientation (Min: 0.0000, Max: 3.7290): 2.0438
 ellipse-length (Min: 0.0000, Max: 832.0300): 534.2343
 ellipse-width (Min: 0.0000, Max: 343.7000): 126.4943
 ellipse-linearity (Min: 0.0000, Max: 5.3000): 4.2234
 mbr-length (Min: 0.0000, Max: 911.3100): 568.6322
 mbr-width (Min: 0.0000, Max: 440.4200): 325.5702
 mbr-linearity (Min: 0.0000, Max: 8.0000): 6.5886
Region PASSED.
------------------------------------------------
```



```
----------------------------------
Firing GAs-border-RWs
Firing between moffett1_2 and moffett1_48
precond-- distance,least: [0.0000,100.0000] 0.4503  true.
* distance, least:
        consistent: [0.0000,30.0000] 0.4503  true
      inconsistent: (75.0000,1000000000.0000) 0.4503  false
      CONCLUSION: consistent
----------------------------------
----------------------------------
Firing RWs-are-parallel-to-GAs
Firing between moffett1_2 and moffett1_48
precond-- distance,least: [0.0000,100.0000] 0.4503  true.
* orientation, parallel:
        consistent: [0.0000,0.5000] 0.0842  true
      inconsistent: (0.8000,1000000000.0000) 0.0842  false
      CONCLUSION: consistent
----------------------------------
```

Figure 10: SPAMEVALUATE: Invoking RTF and LCC schema.

4.3. Verification of the SPAM System

One other tool deserves mention as a simple diagnostic tool. SPAMANALOG aids in the detection of problems with SPAM after it has been compiled into a production system. Since SPAM is a very large and complex production system, as we add new types of knowledge (and thus new types of productions), it is easy to introduce bugs. It is not feasible to trace every production firing of a SPAM run to check for errors, since even a small run can contain as many as 80,000 production firings, while larger runs can number 500,000 or more productions fired [9].

To find mechanical problems in the system, SPAMANALOG examines the logfile produced by SPAM and tallies the productions fired, listed by phase and by individual productions. Also extracted are timing information and any error messages that SPAM may have produced. The result is a simple summary, such as the one in Figure 11, that we use to spot anomalies in the production firings. In addition, even though it is often impossible to directly compare logfiles of different runs (because of their size), it is easy to compare two SPAMANALOG summaries of them. We exploit this property to verify that new control and domain knowledge is not adversely affecting the mechanics of the run.

```
SUMMARY OF PRODUCTIONS FIRED

Production type          # fired
------------------     ----------------
   RTF                      14431
   LCC                      56864
   FA                        6820
   MG                        5133
   SPAM                         5
   other                        2

Production name                                            # fired
------------------                                     ----------------
RTF**GENERAL*check-match*analyze*                             952
RTF**GENERAL*check-match*make-fragment*                       253
RTF**GENERAL*check-match*missing-confidence-threshold*         15
RTF**GENERAL*check-match*under-threshold*                     699
RTF--TW--initialize-TW-attributes                              32
RTF--TW--match-TW-area                                         32
RTF--TW--match-TW-compactness                                  32
RTF--TW--match-TW-curvature                                    32
RTF--TW--match-TW-ellipse-length                               32
      .
      .

***** WARNING : The following productions never fired: *****
RTF**GENERAL*subtask-exit*
RTFSUPP**curved-alignment**prepare-process
RTFSUPP**curved-alignment**queue-process
RTFSUPP**fork-process-1
      .
      .
      .

Timing Information
----------------------------

Init:   User_Time:    0.4000 sec, Sys_Time:  1.0900 sec, Tot_Time:    1.4900 sec
Match:  User_Time:  584.8800 sec, Sys_Time: 10.7000 sec, Tot_Time:  595.5800 sec
RHS:   User_Time: 2599.6000 sec, Sys_Time: 96.3600 sec, Tot_Time: 2695.9600 sec
Load:  User_Time:    6.0300 sec,  Sys_Time:  0.2700 sec,  Tot_Time:    6.3000 sec

Productions Fired: 83255
RHS Actions Performed: 121545
Working Memory Changes: 87199
Shared Memory Usage:  46080000 bytes alloc'd, 7298044 bytes used, 15.84% usage.
```

Figure 11: Example of SPAMANALOG output.

5. Experiments and Results

One tangible result of an image interpretation system should be the generation of a labeled set of segmentations. For SPAM, this labeled set includes the context in which the interpretations were decided upon (the model(s) and their constituent functional-areas), as well as the set of interpretations and related consistency information. However, a task as difficult and complex as aerial image interpretation allows many opportunities to contribute to the body of research knowledge, in addition to simply generating a final result. Evaluation of the effectiveness of different types of knowledge is an interesting as well as important element of our research goals. In this section, we present some experimental results from each of the phases of processing in SPAM. We discuss these results and evaluate the effects of the knowledge acquisition tools presented in the previous section.

5.1. Improving Shape and Spatial Constraints

As seen in Section 3, many of the tools developed for use with SPAM were created with the specific purpose of improving the results emerging from a single processing phase. For example, the program RTFANALYZE allowed us to automatically generate sets of RTF schemas from statistics produced from our airport ground-truth database. Because the RTF phase deals only with local properties of each segmentation, a brute-force approach is tractable. These automatically generated schema were found to improve the classification of certain object classes (e.g., taxiways) when compared with the hand-generated schema. Figure 3 shows these differences. For example, the number of true positives for taxiways increased from 11 to 28. However, the number of false positives also increased from 11 to 46. This trend is true, in general, across all object classes. The issue is that most of the false positives will not be found to be consistent with other hypotheses during SPAM's LCC phase. Therefore, it is more important for a larger number of true positives to proceed past RTF.

Class	# Ground Truth	Auto True Positives	Auto False Negatives	Auto False Positives	Auto True Negatives	Hand True Positives	Hand False Negatives	Hand False Positives	Hand True Negatives
runway	4	3	1	0	163	0	4	0	163
taxiway	39	28	11	46	82	11	28	11	117
road	2	1	1	20	145	0	2	3	162
terminal-building	9	4	5	54	104	2	7	23	135
hangar-building	17	11	6	56	94	5	12	12	138
parking-apron	0	0	0	55	112	0	0	59	108
parking-lot	0	0	0	74	93	0	0	98	69
grassy-area	45	43	2	57	65	35	10	14	108
tarmac	6	3	3	82	79	1	5	36	125

Table 3: Comparison of RTF confusion matrices for hand versus auto generation for San Francisco.

From evaluating the results of running SPAM on several different airport scenes, it became clear that knowledge in LCC was incomplete. This was most evident when examining functional-area results, as it was easy to see that certain object classes were incorrectly supporting (or not supporting) one another. For the LCC phase, quantitative result evaluation is more complicated because constraints are not universally true. For example, though distance constraints exist between hangar-buildings and roads (as classes), there may be no such relationship between a specific hangar-building/road pair. A ground-truth database of constraints to permit such an evaluation would be tedious and time-consuming to generate (though not impossible). In lieu of generating such a database, we designed several experiments to assist us in evaluating the effectiveness of the LCC rules. Two of these experiments were:

- Replacing the output from RTF with a set of perfect interpretations;
- Supplementing the output from RTF with a set of perfect interpretations.

The set of perfect interpretations can be easily generated from the ground-truth. We can qualitatively evaluate the results of these experiments using our knowledge acquisition tools while also refining the existing knowledge base. We can then quantitatively evaluate the results of rerunning the SPAM system.

The first experiment helped to detect problems with lack of support between correct hypotheses (false negatives). Missing support would indicate either a lack of constraints or incorrect (too narrow) bounds on the existing constraints. SPAMEVALUATE and LCCCHK were used to evaluate proposed constraints as well as to locate and fix problems with existing constraint bounds.

The second experiment aided in identifying incorrect hypothesis support (false positives). As stated previously, SPAM assumes that *consistency* between two interpretations is determined by the action of multiple constraints. Therefore, it is expected that there will be many constraints falsely satisfied, and that they will be distributed randomly among the available hypothesis classes. However, if a rule has a very permissive bound, then many false positives occur, and the constraint poorly discriminates one class from another. Again, both SPAMEVALUATE and LCCCHK were used to identify useless constraints and to tighten bounds on permissive constraints.

In addition to manual modifications to the knowledge base, we used FAANALYZE to semi-automatically generate constraints for the LCC phase. FAANALYZE generates all possible geometric constraints and performs only simple pruning of the results. A significant portion of these constraints provide very little discrimination between consistent and inconsistent hypotheses, though several missing constraints were found. Therefore, creating LCC schema is still a task for a knowledgeable user, but FAANALYZE is still useful as an advisory tool, supplying the range of expected values once a geometric constraint has been selected.

5.2. Functional Groupings

Viewing the results of FA from a number of airports using SPAMEVALUATE revealed several problems, the most apparent of which were:

- Large functional-areas — The generated functional-areas covered large portions of the scene and included many regions.
- Multiple interpretations — Within a functional-area, a given region could have multiple conflicting interpretations.

Discussion of and solutions to these problems are reviewed below.

5.2.1. Focusing Functional-Areas

The ideal functional-area should include only the seed (or, generating) region and those regions consistent with the seed that surrounding it. SPAM requires small functional-areas for several reasons. One purpose of the functional-area is to provide an object whose set of constituents are relevant to one another, i.e., they exert some constraint on most of the other members of that set. If a functional-area is too large then bounds on constraints between members of that functional-area will tend to be less precise. Additionally, a large number of functional-area elements makes the process of deciding ambiguous interpretations computationally more difficult (see Section 5.2.2).

Using knowledge that was collected before the creation of our knowledge acquisition tools, the functional-areas generated by SPAM were too large, both in size and in number of elements. Some of this was corrected as a result of using these tools to improve both RTF and LCC results. However, enough problems remained that we felt it necessary to develop a method for focusing the application of constraints to local regions around our current area of interest.

A natural solution to this problem was to allow preconditions in local-consistency rules in order to force each rule to focus on smaller portions of the scene. For example, we use a maximum distance test to prune away regions that are being considered for orientation tests. Because the functional-area is computed from the local-consistency information, this addition reduces the number of interpretations considered and, therefore, the size of the functional-area.

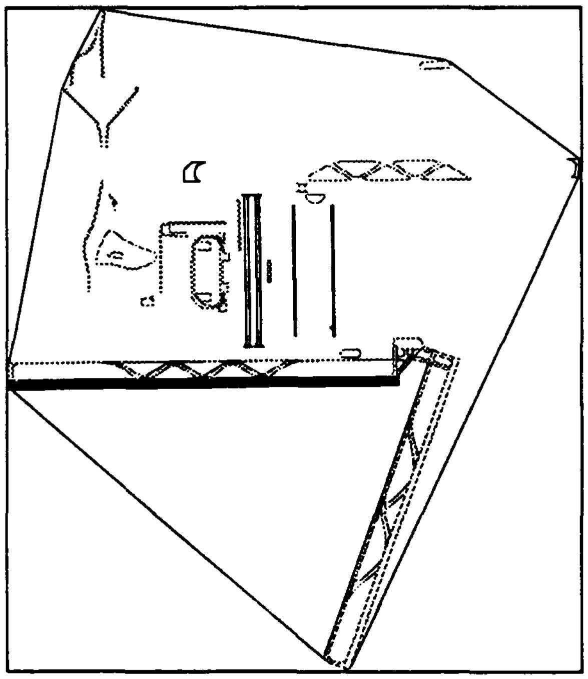


Figure 12: Functional-Area from Dulles International.

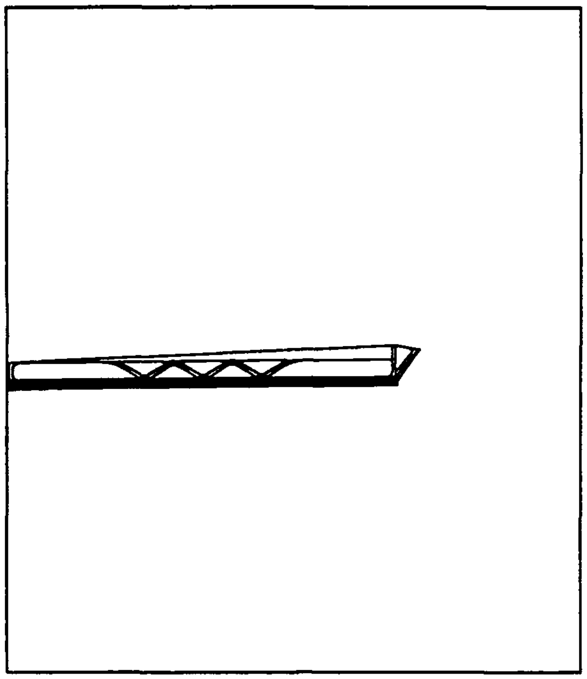


Figure 13: New Functional Area after Preconditions.

Example functional-areas, before and after addition of preconditions, are shown in Figures 12 and 13, respectively. The seed for these two functional-areas is the same (the runway region highlighted in bold), and the interpretations that are incorrect or ambiguous are shown in grey. The change in size is obvious, and one can see that not only is the new functional-area more focused, but it contains no incorrect or ambiguous interpretations when compared to the ground-truth. As expected, the preconditions are providing a context around each interpretation within which we can apply the spatial constraints.

5.2.2. Ambiguous Interpretations

Another problem we addressed was the resolution of ambiguous interpretations within a functional-area. Previously, we would delay resolution of such conflicts until MODEL phase, making the model-generation process more complicated. Making these decisions in FA seems more intuitive, as functional-areas emerging from this phase are, as a result, internally consistent.

The context provided by the functional-area is used to decide between two or more competing interpretations. Again, the solution that seemed most natural was to use more complicated LCC constraints. An example of such a constraint is *tarmacs are close to and located at the ends of taxiways*. These types of constraints could be implemented as functions of the binary constraints already used in LCC. This suggests that a function involving a count of the number of positive and negative links could be an effective method of determining which of several hypotheses is most supported. This function should have the property that, when evaluated over all the supporting hypotheses, fewer multi-constraint successes should receive a higher confidence than many single-constraint successes.

We chose a function that totaled all positive links from unambiguous hypotheses (regions with a single interpretation in the context of the functional-area) for each ambiguous interpretation (regions with multiple interpretations within the context of the functional-area). Those unambiguous hypotheses supporting the conflict with more than a single satisfied constraint receive extra weight in the confidence computation. The interpretation with the most weight remained in the functional-area; any competing interpretations were filtered out.

Table 4 shows statistics compiled across all the filtered functional-areas generated from a single run. The chart shows the four airport functional-area types, broken down into their respective elements. Each element-type is followed by four columns showing, in order, the number of correct hypotheses of that type found in functional-areas, the number incorrect, and then the results of the filtering (number of correct/incorrect hypotheses filtered). The numbers indicate that, in most cases, the incorrect hypotheses are being filtered out more often than correct ones. For instance, every incorrect hypothesis of the runway functional-areas is filtered, and only two correct hypotheses are removed. This means that the filtering algorithm and the consistency information it uses are generally working well. A specific observation should be made with regard to terminal-building functional-areas. From the table it can be seen that parking-aprons within terminal-building functional-areas seem to be filtered more or less randomly. This is due in part to there being such a small ratio of correct to incorrect terminal-building functional-area seeds. This is the fault of the consistency information for terminal-building functional-areas, not of the filtering algorithm.

5.2.3. Relative Shape Constraints

A less successful experiment was the development of relative shape constraints. As described in Section 3.4, the relative shape constraints endeavor to normalize the shape-attribute values of regions of a functional-area, using the seed of the functional-area. This allows tighter bounds on the attributes, since they will adjust themselves to larger or smaller airports automatically. We expected these new constraints would help in several instances. We wanted to use them to weed hypotheses from functional-areas, or to weed internally inconsistent functional-areas entirely. In addition, these constraints could be used to filter hypotheses *before* the LCC phase runs, reducing the amount of work the LCC phase must do.

The relative shape constraints turned out to be very weak. Experiments showed that while they were capable of some weeding, they basically duplicated a small subset of the results we were already obtaining from the LCC phase. As for detecting incorrect functional-areas, the

FA Type	Element Type	Correct Hypotheses	Incorrect Hypotheses	# Correct Hypotheses Filtered	# Incorrect Hypotheses Filtered
runway	runway	3	0	0	0
	taxiway	24	1	0	1
	grassy-area	20	6	2	6
	tarmac	0	19	0	19
hangar	hangar-building	75	319	0	0
	road	79	65	3	21
	parking-apron	11	219	1	117
	tarmac	19	236	0	140
terminal	terminal	2	105	0	2
	road	102	104	10	29
	parking-apron	16	335	14	281
	parking-lot	43	541	0	61
road	road	128	47	0	0
	grassy-area	94	126	0	29

Table 4: Dulles functional-area summary.

differences between correct and incorrect functional-areas were very slight, and too unreliable to quantify. However, we did note that relative shape-constraints are much faster than computing LCC-constraints, so running the FA phase with relative constraints before firing the LCC phase might in fact speed the system up with very little change in the results. We plan to experiment with this in the future.

5.3. Results for Model Generation

The final phase of SPAM must produce a consistent scene model by merging functional-areas while resolving any ensuing conflicts. Multiple models should be generated when competing models of the same scene are radically different in layout. Model generation is a hard problem, conceptually and combinatorially; the strictly algorithmic approach, generating all possible scene models, can be shown to be NP-complete (see Section 5.3.2). This section outlines three different methods that we have used for generating final scene models from functional-areas. These are:

1. Generating a tiling of the scene;
2. Generation of maximal cliques from a graph;
3. Using heuristic search.

We are experimenting with each of these methods, though the heuristic search method currently shows the most promise.

5.3.1. Tiling

One of our approaches to producing a scene model involves generating a tiling of the scene in question. Such an algorithm tries to choose those functional-areas that maximize the amount of the scene that is covered. This would be a trivial problem if it weren't for the fact that functional-areas overlap, and interpretations for those regions in the areas of overlap can conflict.

These conflicts must either be avoided, or resolved intelligently, as they determine the overall goodness of the generated model. Therefore, constraints are applied which allow the system to intelligently choose which functional-areas can coexist.

The algorithm works as follows. Initially, the model is empty. A model *seed* (an initial functional-area) is chosen, either automatically or by the user, and consecutive functional-areas are added to this seed based on maximizing the number of new regions, minimizing the number of hypothesis conflicts, and maximizing support from existing functional-areas. The process terminates when a given percentage of the input regions have been used in the model.

This procedure efficiently generates a single model which is very dependent on the quality of the initial seed functional-area. We believe this dependence could be lessened by creating better functional-area constraints. Further development would emphasize embedding this algorithm in a framework that would allow the ability to generate and rank multiple models.

5.3.2. Maximal Cliques

Another approach to model generation views each functional-area as a node in a graph, with arcs existing based on the compatibility of two functional-areas. Compatibility can be computed by applying constraints, such as those used in Section 5.3.1, to functional-area pairs. Once the arcs have been computed, models can be extracted from the graph by searching for maximal cliques, or by looking for cliques of a particular size. Algorithms for these problems exist, but their worst-case performance is known to be exponential in the number of inputs. In this case, the number of models generated can be exponential in the number of functional-areas. For our experiments, only the smallest sets of functional-area results could be used to generate models (no more than 40 or 50 functional-areas). Modifications to this method, such as adding domain knowledge to constrain the search for cliques, or reducing the numbers of nodes in the graph by merging functional-areas of the same type, could help make the process tractable.

5.3.3. Heuristic Search

Our most recent research in model generation uses heuristics to constrain which sets of functional-areas can coexist in the same model. A best-first search is performed in the space of possible functional-area groups. Conflicts between competing hypotheses from different functional-areas are enumerated, but are currently not resolved.

The key to directing the search for models efficiently and intelligently is the set of heuristics used. No attempt is made to rank or order the heuristics. Instead, all heuristics contribute equally to the functional-area's score, some in a positive manner and others negatively. The score improves if the functional-area adds new regions and/or support to the current model, is very compact, or covers a lot of area. The score is reduced if conflicts are added to the model. The score is normalized according to the number of regions in the functional-area in an attempt to allow functional-areas of varying sizes to participate equally in model-generation. Other heuristics include the density of the entire model, and the density of the conflicts. A tight group of conflicts is believed to be better than one with conflicts spread out all over the scene.

All the model heuristics discussed thus far are domain-independent. There are instances where one would like to bring domain-dependent constraints to bear, such as constraints between functional-areas, much like the LCC phase evaluates consistency between pairs of regions. Examples of such constraints are *terminal-buildings are centrally located with respect to the runways* (in fact, there are functional reasons for the control-tower or terminal-building area to be more or less equidistant from each runway) and *hangar-buildings are not centrally located.*

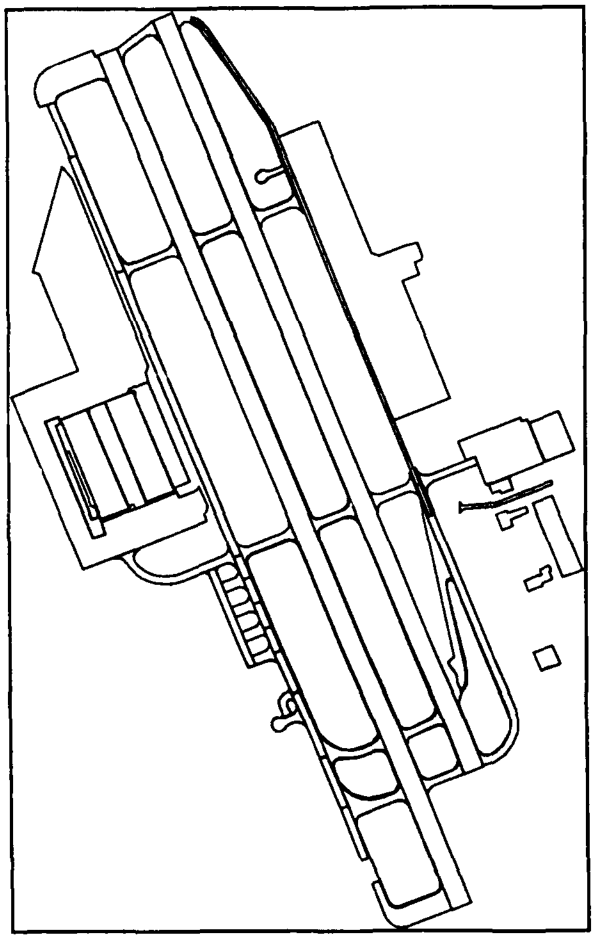


Figure 14: Model of Moffett AFB.

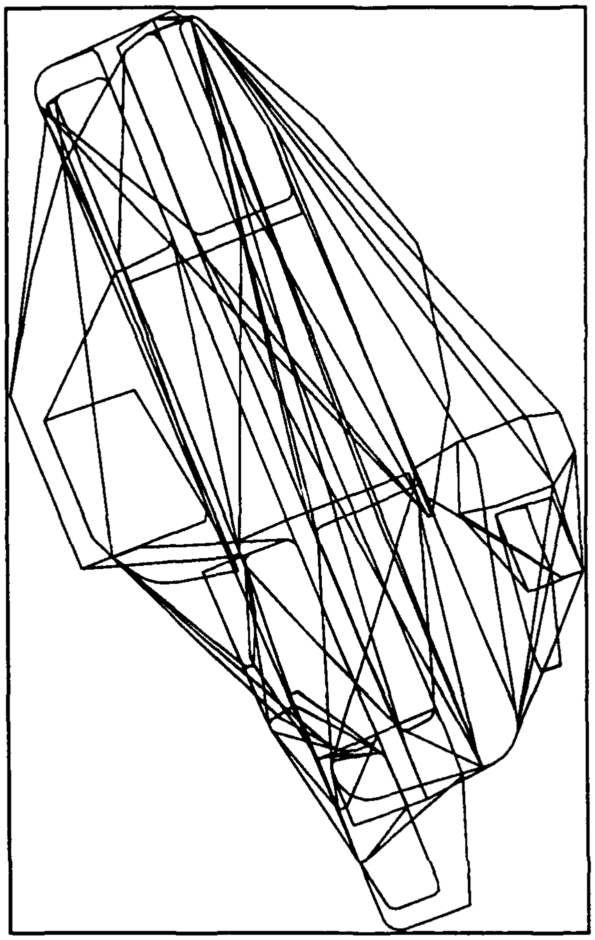


Figure 15: Functional-Areas in the Model.

Our experiments with these constraints indicate that they are not good predictors and so cannot be used to intelligently generate and/or fix models, though they can be used to evaluate existing models. Therefore, our current scheme generates multiple models (without using scene dependent knowledge), then allows domain-dependent constraints to be used to rank the models produced. Further exploration of these types of constraints is material for future work.

• •

6. Task-Level Parallelism

The production system programming model used by SPAM allows knowledge to be added easily. However, large production systems like SPAM continue to suffer from extremely slow execution times, which limits their utility in practical applications, as well as in research settings. Most efforts at speeding up these systems have focused on match or knowledge-search parallelism in production systems. Although good speed-ups have been achieved in this process, the total speed-up available from this source is not sufficient to alleviate the problem of slow execution in large-scale production system implementations. Such large-scale tasks can be expected to increase as researchers develop increasingly more competent rule-based systems.

Another area of our research, done in conjunction with the Production System Machine group at Carnegie Mellon, has focused on task-level parallelism (TLP) [10, 11], which is obtained by a high-level decomposition of the production system. We will describe some timing results from the original, Lisp-based SPAM system, then show our problem decomposition, and finally show the resultant performance improvements obtained using task-level parallelism running on a 16 processor Encore Multimax.

6.1. Diagnostic Timing Results

Table 5 gives statistics for the run-time and number of production firings for each interpretation phase in SPAM for *San Francisco International Airport (SF)*. These statistics are representative of those obtained from other data sets. It is interesting to note that LCC and FA phases account for most of the overall time in a complete run. Further, within these phases, much of the rule evaluation is performed outside of the OPS5 production system using external processes. For example, FA spends much of its time doing evaluation outside of OPS5. RTF, on the other hand, spends most of its time within the traditional OPS5 evaluation model and consumes less time than FA, even though it executes a comparable number of productions. It is also clear from this table that the application of spatial constraints in LCC makes it by far the most expensive phase in terms of amount of time spent, number of productions, as well as number of production firings.

SPAM Phase	RTF	LCC	FA	MODEL	Total
Total CPU Time (hours)	1.5	144.5	7.3	0.71	154.01
Total Productions Fired	11274	185950	10447	3085	210756
Effective Productions/Second	2.08	0.357	0.397	1.20	0.380
Total Hypotheses	466	N/A	44	1	N/A

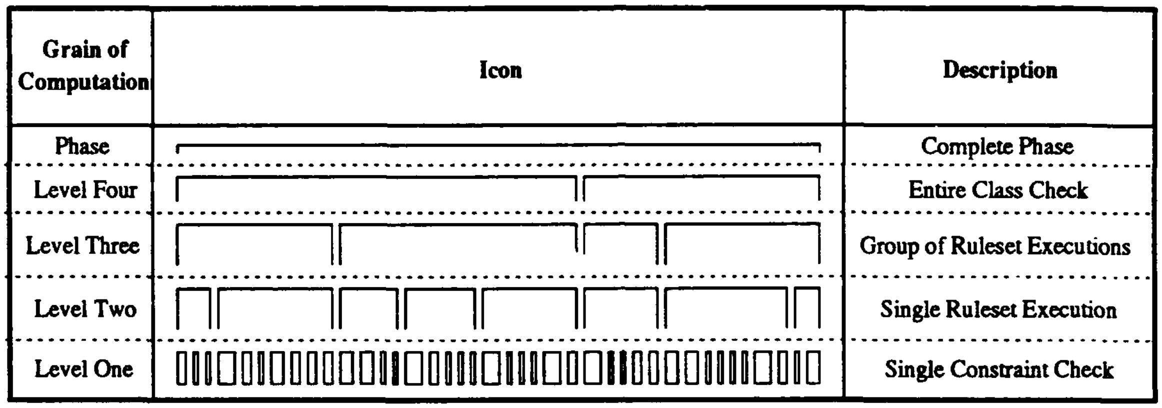
Table 5: San Francisco Airport (log #63)

During the LCC phase, knowledge of the structure or layout of the task domain (i.e. airports or suburban housing developments) is used to provide spatial constraints for evaluating consistency among fragment hypotheses. For example, *runways intersect taxiways* and *terminal buildings are adjacent to parking apron* are examples of the kinds of constraints that are applied to the airport scene segmentation. It is important to assemble a large collection of such consistency knowledge because the results of these tests are used to assemble fragment hypotheses found to be mutually consistent as contexts for further interpretation within the functional area phase.

Just from the raw statistics, it seems clear that the LCC phase should be our candidate for applying parallelism. Another rationale for this approach is the observation that this phase has the largest potential for growth. If a single new scene primitive is added within the RTF phase, many constraints may be added in the LCC phase in order to describe the spatial relationships (and constraints) between each of the other primitives. For these reasons, we believe that as new knowledge is added to the existing SPAM system, the proportion of time spent in LCC phase can only increase.

6.2. Decomposing the LCC Phase

As a result of this preliminary analysis we decided to focus our initial efforts on the parallel implementation of the LCC phase[4]. The LCC phase applies geometric knowledge (constraints) from the selected domain to the set of interpretations made from the dataset. This application of geometric knowledge can be logically decomposed into several levels, where the tasks within each level are independent and can be performed in parallel. This is illustrated in Figure 16.



Grain of Computation	Icon	Description
Phase		Complete Phase
Level Four		Entire Class Check
Level Three		Group of Ruleset Executions
Level Two		Single Ruleset Execution
Level One		Single Constraint Check

Figure 16: Levels of processing in SPAM LCC.

In order to choose the right level of decomposition at which to parallelize the SPAM LCC phase, we instrumented the SPAM system to obtain measurements at each level for the number of tasks and their run-time average, standard deviation, and coefficient of variance. The results of these measurements for each of the San Francisco airport dataset are presented in Table 6.

Level	Average time per task (sec)	Standard deviation (sec)	Coefficient of variance	Number of tasks
Level 4	875.27	525.92	0.601	9
Level 3	65.65	29.51	0.449	120
Level 2	20.90	8.48	0.406	377
Level 1	0.489	0.0782	0.159	16104

Table 6: Average, standard deviation and coefficient of variance for SF.

Using information from Table 6 the appropriate level of granularity can now be chosen. For Level 4, the task to processor ratio is smaller than one, so we immediately rejected pursuing

[4]Since the analysis is performed using the original, expensive Lisp-based SPAM system, we have extracted a representative subset of the airport dataset to drive the analysis.

parallelism at this level. Levels 3 and 2 are very similar to each other in that they have enough tasks, their variances are not large, and the task granularities are much larger than the expected task management and communication overheads. Both levels, therefore, seemed to us to be worthwhile candidates. Level 3 seemed somewhat more desirable as less effort appeared to be required of us to achieve amounts of parallelism similar to that available in Level 2.

Level 1 was rejected for several reasons. First and most importantly, the additional effort involved in decomposing the system at the granularity of Level 1 would not allow us to achieve any more parallelism than at Level 2 or 3 because of the limitation on the number of processors. Second, the task granularity is much smaller and thus closer to the overheads for task management and communication than any of the other levels. Finally, the task to processor ratio is on the order of 1000. This can have a detrimental effect due to the initialization overhead. Our conclusion, then, was to exploit parallelism at the granularity of Levels 2 or 3.

6.3. Results from Using Task-Level Parallelism

Results for parallelizing Levels 2 and 3 are shown in Figure 17. The speedups are computed against a baseline version, which represents an optimized uniprocessor implementation of the SPAM LCC phase. It is interesting to note that this uniprocessor baseline version provides approximately a 10 to 20 fold speedup over the original Lisp-based implementation for the LCC phase.

The results of applying task-level parallelism are shown in Figure 17. Curves are shown for the San Francisco dataset, as well as for two other airport datasets. The speed-up curves show near linear speed-ups for both levels of decomposition. The speed-ups within a level are almost the same among the three airport datasets. The maximum speed-up achieved using 14 processors is 11.90 fold in Level 3 and is 12.58 fold in Level 2.

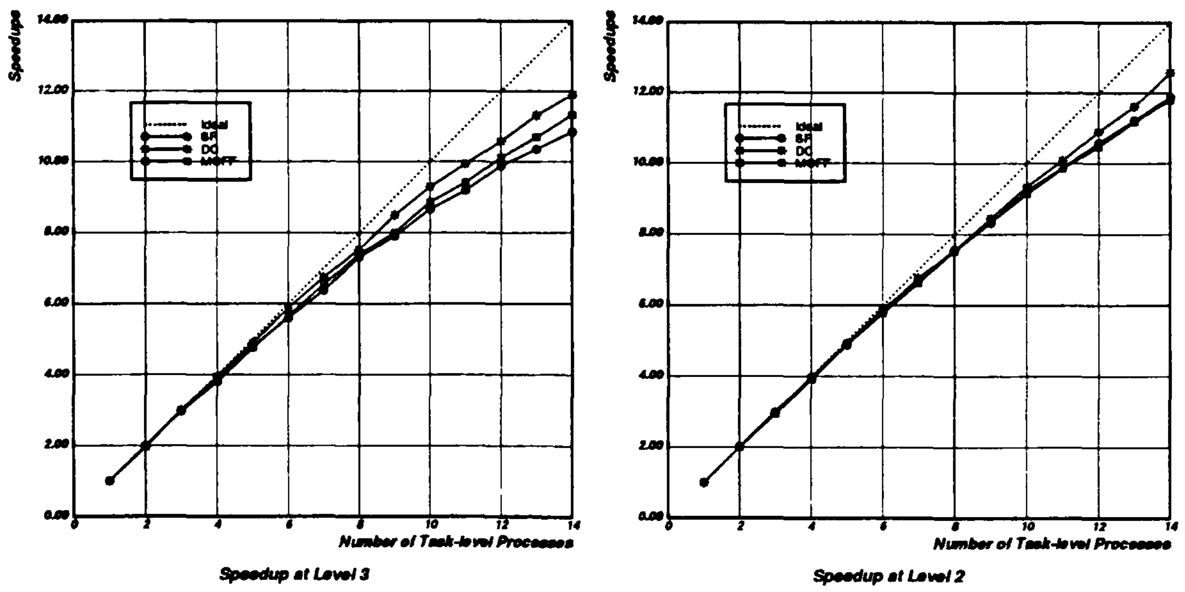


Figure 17: Speed-ups varying the number of task-level processes.

For match parallelism, the theoretical maximum speed-up that can be obtained is limited according to the percentage of total execution time spent in match. As SPAM spends less than 50% of its time in match, speed-ups due to match parallelism are limited to a factor of 2 or less. This is exactly what is observed.

We believe that the potential for additional speed-ups in SPAM from task-level parallelism is quite high; an expectation of 50 to 100 fold does not seem unreasonable, because:

- The tasks within any of the LCC decompositions are independent of one another;
- Several hundred tasks are available in Level 2;
- The task queue management overheads measured for Level 2 and Level 3 are very low, especially with respect to the task granularity, and thus are not a factor.

Although our scheme of parallelization has been presented in the context of a non-match-intensive system, the scheme is applicable to match-intensive systems as well. In match-intensive systems, match parallelism will make a substantial contribution to the speed-ups.

6.4. Latest Work with Task-Level Parallelism

Our most recent work has been concentrated in three areas. These are:

1. Investigation of the causes of non-linear speed-ups;
2. Merging the results at the end of a run;
3. Preliminary use of the network-shared memory server.

Each of these topics are discussed below.

6.4.1. Non-linear Speed-ups

With the extremely low overheads associated with task management, the speed-ups obtained by the SPAM/PSM system were somewhat lower than expected, although the system produced good speed-ups (12.5 fold using 14 processors). This was particularly obvious when using higher numbers of processors. To discover where the extra processing time is being spent, we have been doing a detailed instrumentation of the SPAM/PSM system to measure contention on shared resources (such as the task queue and the memory allocator). However, these instrumentation results revealed that there is little, if any, contention for shared resources. We have since re-programmed the system to eliminate any remaining contention and this resulted in no improvement to the speed-up numbers. We are currently investigating other effects that could cause a degradation in system performance. This currently includes determining whether paging is contributing significantly to our loss in performance, and examining the behavior of various MACH system calls in a parallel environment.

6.4.2. Result Merging

Our second area of work centered on further development of the SPAM/PSM system as a tool for SPAM researchers. The previous version of the system would compute results (in parallel) and place them in local working-memories, but would then throw these memories away when the computation exited. Originally, the decision to ignore this (and other similar problems) allowed us to concentrate on decomposition and speed-up issues. Now that it has been demonstrated that good speed-up results are obtainable, we must address the engineering issues to make the system usable. To this end, we have added code to the SPAM/PSM system to concurrently merge results when the run terminates. A nice property of our solution is that it efficiently reuses processors, thereby avoiding the introduction of a serial bottleneck at the end of a run. Other engineering issues that will be addressed are multiple invocations of task-level processing (the current system allows only one initialization/parallel execution cycle), space problems (we need to use memory more judiciously for some common data structures), and software problems (integration of newer versions of SPAM and PSM code).

6.4.3. Network-Shared Memory System

The results reported above show that large amounts of parallelism can be exploited in SPAM, and thus, significantly larger numbers of processors could be employed in exploiting the parallelism. The limitations on the number of processors on the Encore Multimax led us to our third area of work, namely the investigation of the *shared virtual memory* system. A shared virtual memory system can provide a single virtual address space among multiple machines, allowing the programmer to view the networked machines as a loosely-coupled multiprocessor. Our local computing environment has two separate Encore multimax machines, each with 16 processors. Recently, the *shared memory server* became available on these machines, providing a 32 processor (16 from each Encore) virtual shared memory machine.

The latency across the two Encores with the virtual shared memory is reported to be 50 ms, much lower than the granularity of the SPAM/PSM tasks. Furthermore, the SPAM/PSM tasks are independent, with the processes requiring minimal communication through the task queue. Therefore, the SPAM/PSM system appeared to be an ideal candidate for experimenting with the shared virtual memory system.

Introducing shared virtual memory in the SPAM/PSM system turned out to be more complex than our initial expectation. In the shared virtual memory system, the programmer has to be sensitive to the allocation of data-structures to pages to avoid contention. This contention problem was made acute due to *false contention*, i.e., two or more processes across the Encores contending for objects located on the same page though not shared between them. At first, no attention was paid to this problem — however, the overhead incurred from constantly page faulting across the network due to false contention brought our system to a halt just during the initialization. Two separate approaches were employed to solve this problem. We organized our data-structures in the address space in order to eliminate the contention across Encores. The designers of the shared virtual memory system proceeded to provide some optimizations and heuristics in the netmemory server to minimize the amount of data sent over the network to service a page fault. For example, instead of shipping a full 8K page, the server ships only small, 64-byte segments of the page that has been modified.

After the elimination of the false contention problem, and the introduction of other optimizations, real speed-ups were possible. The speed-up results are shown in Figure 18. The following observations can be made about these results:

1. Two separate speed-up curves are shown in Figure 18. The first one was obtained from the shared virtual memory system. The second was obtained from the pure task-level parallelism system, i.e., the system without the shared virtual memory (the speed-up curve from this system is indicated in Figure 18 as Pure TLP.) The comparison of these two curves shows that real speed-ups are indeed possible with the shared virtual memory system — underscoring the usefulness of the shared virtual memory system for large applications. However, these speed-ups came only after many rounds of optimizations in our system and in the shared virtual memory server.

2. As we can see from Figure 18, the speed-ups for the shared virtual memory system are close to the pure task-level parallelism system, as long as all the processes are running on a single Encore. As soon as we add a process on the remote Encore, we see an abrupt change in the curve — which produces a translational effect on the curve. This translational effect is equivalent to the loss of about 1.5 processors. This performance hit comes from the overheads of communication across the network.

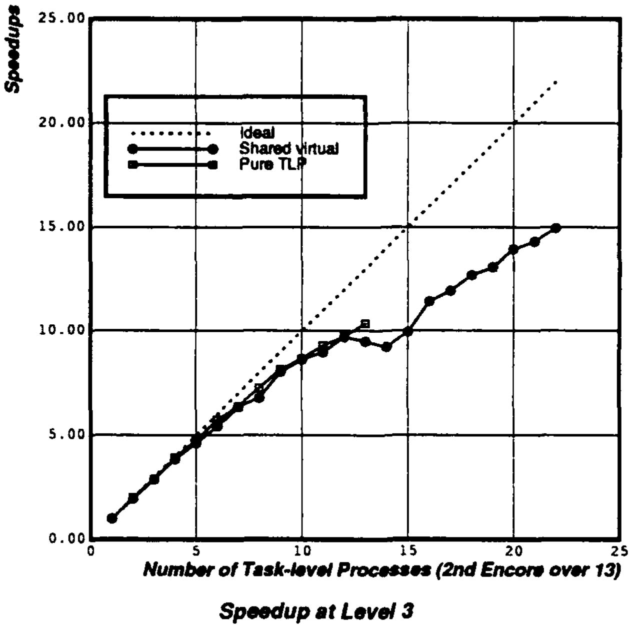


Figure 18: Speed-ups varying the number of processes using the virtual shared memory server.

3. We were only able to provide results for 22 processors — 13 on the first Encore and 9 on the second. Our application placed severe demands on the MACH kernel, preventing us from using all the processors. These robustness issues are being addressed by the designers of the shared virtual memory system.

4. In our final optimized system, only a single task queue is present. The contention for this task queue is minimal. Separate experiments were performed on these task queues, which indicated that introducing separate task queues (one for each Encore) would not change the results.

Further investigation of the shared virtual memory system continues. Also, as reported previously, our experiments with the network shared-memory system provided real speed-ups (15 fold using 22 processors) and many useful experiences. The software environment provided on the Encore machines, however, has been undergoing revision and is somewhat unstable. Because we feel that the distributed shared-memory paradigm has significant potential, we've been working closely with researchers in the MACH operating system group to help them debug and tune their software. Most of this work has involved running the netmemory TLP system with various new pieces of system software (kernel, network memory server) to allow the maintainers to measure the software's performance. Once this software environment stabilizes, we can continue analysis and experimentation on the SPAM/PSM netmemory system.

7. Open Issues

SPAM supports a variety of research activity within the context of image understanding. It is a research vehicle for our knowledge acquisition work, experience with systems integration, and further research in task-level parallelism [11, 9]. As such, there are many promising areas of future work. Several of these are outlined below.

As with any system, testing on many cases aids in finding implicit assumptions. More airport examples would not only help in refining SPAM's knowledge base, but this would allow exploration of categories within the airport domain. For example, it may not be as effective to have a "general" airport knowledge base as it would to have a separate knowledge base for each one of *military*, *international*, and *regional* airports.

In addition to other airports, different segmentation methods will likely provide SPAM with new challenges. We have, at various times, experimented with using segmentations obtained from automatic methods, such as some of the feature extraction systems being developed in our group [12, 13, 14, 15]. Issues include how much of the knowledge has to be tailored according to the source of the segmentation, as well as what effects are produced by errors in the segmentation. Such feature extraction systems could also be used in the later phases of SPAM as domain experts, assisting in the disambiguation of conflicting hypotheses.

Expanding some of our work in automatic constraint generation for SPAM, such as was done with RTFANALYZE and FAANALYZE, is another promising area of future research. Applying clustering techniques may allow the system to automatically filter out constraints that result in little or no discrimination among competing hypotheses. One could augment such a system with interactive capabilities so that it could suggest rules to a user and allow that user to reject or refine them.

Much work has been done on performance evaluation within the SPAM system. Until recently, this work has been focused on manual or interactive evaluation. An diagnostic system that would evaluate SPAM's results and automatically (or semi-automatically) correct or suggest rules would be valuable. This is appealing for several reasons:

- There are large numbers of constraints, and the interactions between constraints must be considered.
- A consistent approach to rule addition/modification could help to keep SPAM's knowledge base more consistent.
- Machine learning efforts, like those suggested above, can be focussed in the areas of greatest need, instead of considering all data for all possible cases.

In addition to improving the quality of the functional-areas produced by SPAM, we need to intelligently evaluate the coherence of each functional-area, as well as to compare them to one another. One method for doing this is to examine the support for each hypothesis and compare it to a "randomly" supported hypothesis. If this or other methods are found to be effective, functional-areas can be ranked, unpromising functional-areas can be excised, and model-generation can proceed based on this ranking.

Finally, there is a great body of machine learning research, and work on validation and verification is becoming more commonplace [3]. However, little research has been done in combining these two areas. We are now beginning preliminary investigations in this area.

8. Conclusions

Our ongoing research in the automated analysis of complex aerial imagery relies heavily on our ability to provide spatial and structural constraints, encoded as rules, to SPAM, our knowledge based interpretation system. Such knowledge can be expected to come from a variety of sources, including site/architectural design rules, map databases, and human experience.

Large-scale knowledge-based vision systems require specialized tools for both knowledge acquisition and result evaluation. As we have seen, this research has focused on the interactive acquisition of spatial and functional knowledge as well as on fully automated techniques. We have shown preliminary results based on our experiments with airport scenes and also demonstrated the importance of manually compiled ground truth scene segmentations to support rigorous performance analysis tools. The ability to generate accurate evaluations of system performance guides our research along paths that are likely to prove most productive.

Issues in system support for the large computational resources required for high-level image understanding require parallelism from both language support (CParaOPS5), and hardware including shared memory and distributed memory multiprocessors. This means that research at the frontiers needs for be vertically integrated, pushing both the computational science as well as the basic image interpretation domain.

Further research is needed in basic computer vision and image understanding, the architecture of knowledge-based systems, and their integration with spatial databases. We believe that our research addresses some of the most important issues in these areas and that we are progressing toward the development of competent automated scene analysis systems.

9. Acknowledgements

We are grateful for the efforts of our COTR, Dr. Abraham Waxman, in many discussions concerning basic research directions and technology transfer. We also acknowledge the support of Lt. Col. Bob Simpson and Dr. Rand Waltzman of ISTO, Defense Advanced Research Projects Agency (DARPA) for the funding of this research.

Wilson Harvey, Matthew Diamond, and Milind Tambe contributed technical material to this final report. Other members of the Digital Mapping Laboratory helped provide a congenial working environment.

10. Bibliography

1. McKeown, D.M., Harvey, W.A. and McDermott, J., "Rule Based Interpretation of Aerial Imagery", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 5, September 1985, pp. 570-585.

2. David McKeown, Aviad Zlotnick, Frederic Perlant, Yuan Hsieh, Wilson Harvey, and Matthew Diamond, "Research in Digital Mapping", *Computer Science Research Review*, Vol. 88/89, 1988/1989, pp. 39-63.

3. Gupta, U., *Validating and Verifying Knowledge-Based Systems*, IEEE Computer Society Press, 1991.

4. Matsuyama, T.,, "A Structural Analysis of Complex Aerial Photographs", Tech. report, Department of Electrical Engineering, April 1980, Ph.D Thesis

5. Shang-Shouq Vincent Hwang, *Evidence Accumulation for Spatial Reasoning in Aerial Image Understanding*, PhD dissertation, University of Maryland, 1984.

6. Huertas, A., Cole, W., & Nevatia, R., "Using Generic Knowledge in Analysis of Aerial Scenes: A Case Study", *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., August 1989, pp. 1642-1648.

7. Strat, T., & Smith, G., "Core Knowledge Systems: Storage and Retrieval of Inconsistent Information", *Proceedings of the DARPA Image Understanding Workshop*, Morgan Kaufmann Publishers, Inc., April 1988, pp. 660-665.

8. McKeown, D.M., Harvey, W.A., Wixson, L., "Automating Knowledge Acquisition For Aerial Image Interpretation", *Computer Vision, Graphics and Image Processing*, Vol. 46, No. 1, April 1989, pp. 37-81.

9. Harvey, W., Kalp, D., Tambe, M., McKeown, D. and Newell, A., "The Effectiveness of Task-Level Parallelism for Production Systems", *Journal of Parallel and Distributed Computing*, Vol. (to appear), December 1991.

10. Wilson Harvey, Dirk Kalp, Milind Tambe, David McKeown, Allen Newell, "Measuring the Effectiveness of Task-Level Parallelism for High-Level Vision", *Proceedings of the DARPA Image Understanding Workshop*, Morgan Kaufmann, May 1989, pp. 916-933.

11. Harvey, W., Kalp, D., Tambe, M., McKeown, D. and Newell, A., "The Effectiveness of Task-Level Parallelism for High-Level Vision", *Proceedings of the 2nd ACM/SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, March 1990, pp. 156-167.

12. McKeown, D.M. and Denlinger, J. L., "Cooperative Methods for Road Tracking in Aerial Imagery", *Proceedings IEEE Computer Vision and Pattern Recognition Conference*, June 1988, pp. 662-672.

13. R. B. Irvin and D. M. McKeown, "Methods for exploiting the relationship between buildings and their shadows in aerial imagery", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 19, No. 6, November 1989, pp. 1564-1575.

14. Shufelt, J.A., and McKeown, D.M., "Fusion of Monocular Cues to Detect Man-Made Structures in Aerial Imagery", Tech. report CMU-CS-90-194, School of Computer Science, September 1990.

15. Hsieh, Y.C., McKeown, D.M., and Perlant, F.P., ''Performance Evaluation of Scene Registration and Stereo Matching for Cartographic Feature Extraction'', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-14, No. 1, January 1992, pp. to appear.